JWT Authentication

- Configuration Example
- Configuration Options
  - claims
    - Syntax
    - Nested Claims
  - clientConfig
    - clientConfig.tls
      - Storing secret values in Kubernetes secrets
  - jwksFile
    - JWT Header Key ID
  - jwksUrl
    - JWT Header Key ID
    - JWT Issuer Claim
  - signingSecret
    - Storing secret values in Kubernetes secrets

!!! info "Traefik Hub Feature" This middleware is available exclusively in Traefik Hub. Learn more about Traefik Hub's advanced features.

The JWT middleware verifies that a valid JWT token is provided in the `Authorization` header (`Authorization: Bearer <JWT>`). If the token can't be passed as an `Authorization` header, it can be given as form data or as a query parameter. See the `tokenKey` option for more information.

With no specific configuration, a JWT middleware only validates the signature of a JWT and checks the `nbf`, `exp` and `iat` standard claims (if they are present). Custom claim validation can be configured with Custom Claims Validation.

Configuration Example

```yaml
apiVersion: traefik.io/v1alpha1
kind: Middleware
metadata:
  name: test-jwt
spec:
  plugin:
    jwt:
      signingSecret: my-secret
      forwardHeaders:
        Group: grp
        Expires-At: exp
      claims: Equals(`grp`, `admin`)
```

Configuration Options

Field	Description	Default	Required
`signingSecret`	Defines the secret used for signing the JWT certificates. It is then used by the middleware to verify incoming requests. At least one of `signingSecret`, `publicKey`, `jwksFile` or `jwksUrl` options must be set. (More information here)	""	No
`signingSecretBase64Encoded`	Defines whether the `signingSecret` is base64-encoded. If set to `true`, the `signingSecret` is base64-decoded before being used.	false	No
`publicKey`	Defines the public key used to verify secret signature in incoming requests. In that case, users should sign their token using a private key corresponding to the configured public key. At least one of `signingSecret`, `publicKey`, `jwksFile` or `jwksUrl` options must be set.	""	No

Field	Description	Default	Required
`jwksFile`	Defines a set of JWK to be used to verify the signature of JWTs. The option can either be a path to a file mounted on the API Gateway or directly the content of a JWK set file. At least one of `signingSecret`, `publicKey`, `jwksFile` or `jwksUrl` options must be set. (More information here)	""	No
`jwksUrl`	Defines the URL of the host serving a JWK set. The keys are cached if the HTTP Cache Control allows for caching. At least one of `signingSecret`, `publicKey`, `jwksFile` or `jwksUrl` options must be set. (More information here)	""	No
`forwardAuthorization`	Defines whether the authorization header will be forwarded or stripped from a request after it has been approved by the middleware.	false	No
`tokenKey`	Defines the name of the query and form data parameter used for passing the JWT, for applications that can't pass it in the `Authorization` header. The middleware always looks in the `Authorization` header first, even with this option enabled. This option should only be enabled if the JWT cannot be passed as an Authorization header, as it is not recommended by the RFC.	""	No

Field	Description	Default	Required
`claims`	Defines the claims to validate in order to authorize the request. The `claims` option can only be used with JWT-formatted token. (More information here)	""	No
`usernameClaim`	Defines the claim that will be evaluated to populate the `clientusername` in the access logs. The `usernameClaim` option can only be used with JWT-formatted token.	""	No
`forwardHeaders`	Defines the HTTP headers to add to requests and populates them with values extracted from the access token claims returned by the authorization server. Claims to be forwarded that are not found in the JWT result in empty headers. The `forwardHeaders` option can only be used with JWT-formatted token.	{}	No
`clientConfig.tls.ca`	PEM-encoded certificate bundle or a URN referencing a secret containing the certificate bundle used to establish a TLS connection with the authorization server (More information here)	""	No
`clientConfig.tls.cert`	PEM-encoded certificate or a URN referencing a secret containing the certificate used to establish a TLS connection with the Vault server (More information here)	""	No
`clientConfig.tls.key`	PEM-encoded key or a URN referencing a secret containing the key used to establish a TLS connection with the Vault server. (More information here)	""	No

Field	Description	Default	Required
`clientConfig.tls.insecureSkipVerify`	Disables TLS certificate verification when communicating with the authorization server. Useful for testing purposes but strongly discouraged for production. (More information here)	false	No
`clientConfig.timeoutSeconds`	Defines the time before giving up requests to the authorization server.	5	No
`clientConfig.maxRetries`	Defines the number of retries for requests to authorization server that fail.	3	No

claims

Syntax

The following functions are supported in `claims`:

Function	Description	Example
Equals	Validates the equality of the value in `key` with `value`.	Equals(`grp`, `admin`)
Prefix	Validates the value in `key` has the prefix of `value`.	Prefix(`referrer`, `http://example.com`)
Contains (string)	Validates the value in `key` contains `value`.	Contains(`referrer`, `/foo/`)
Contains (array)	Validates the `key` array contains the `value`.	Contains(`areas`, `home`)
SplitContains	Validates the value in `key` contains the `value` once split by the separator.	SplitContains(`scope`, ` `, `writer`)
OneOf	Validates the `key` array contains one of the `values`.	OneOf(`areas`, `office`, `lab`)

All functions can be joined by boolean operands. The supported operands are:

Operand	Description	Example
&&	Compares two functions and returns true only if both evaluate to true.	Equals(`grp`, `admin`) && Equals(`active`, `true`)

Operand	Description	Example
||	Compares two functions and returns true if either evaluate to true.	Equals(`grp`, `admin`) || Equals(`active`, `true`)
!	Returns false if the function is true, otherwise returns true.	!Equals(`grp`, `testers`)

All examples will return true for the following data structure:

```
{
  "active": true,
  "grp": "admin",
  "scope": "reader writer deploy",
  "referrer": "http://example.com/foo/bar",
  "areas": [
    "office",
    "home"
  ]
}
```

Nested Claims

Nested claims are supported by using a `.` between keys. For example:

```
user.name
```

```
{
  "active": true,
  "grp": "admin",
  "scope": "reader writer deploy",
  "referrer": "http://example.com/foo/bar",
  "areas": [
    "office",
    "home"
  ],
  "user": {
    "name": "John Snow",
    "status": "undead"
  }
}
```

```
John Snow
```

!!! note "Handling keys that contain a '.'"

If the `key` contains a dot, the dot can be escaped using `\.`

!!! note "Handling a key that contains a '\'"

If the `key` contains a `\`, it needs to be doubled `\\`.

clientConfig

Defines the configuration used to connect the API Gateway to a Third Party Software such as an Identity Provider.

`clientConfig.tls`

Storing secret values in Kubernetes secrets

When configuring the `tls.ca`, `tls.cert`, `tls.key`, it is possible to reference Kubernetes secrets defined in the same namespace as the Middleware.
The reference to a Kubernetes secret takes the form of a URN:

```
urn:k8s:secret:[name]:[valueKey]
```

```
apiVersion: traefik.io/v1alpha1
kind: Middleware
metadata:
  name: test-jwt
spec:
  plugin:
    jwt:
      clientConfig:
        tls:
          ca: "urn:k8s:secret:tls:ca"
          cert: "urn:k8s:secret:tls:cert"
          key: "urn:k8s:secret:tls:key"
          insecureSkipVerify: true
```

```
apiVersion: v1
kind: Secret
```

```
metadata:
  name: tls
stringData:
  ca: |-
    -----BEGIN CERTIFICATE-----
    MIIB9TCCAWACAQAwgbgxGTAXBgNVBAoMEFF1b1ZhZGlzIExpbWl0ZWQxHDAaBgNV
    BAsME0RvY3VtZW50IERlcGFydG1lbnQxOTA3BgNVBAMMMFdoeSBhcmUgeW91IGRl
    Y29kaW5nIG1lPyAgVGhpcyBpcyBvbmx5IGEgdGVzdCEhITERMA8GA1UEBwwISGFt
    aWx0b24xETAPBgNVBAgMCFBlbWJyb2tlMQswCQYDVQQGEwJCTTEPMA0GCSqGSIb3
    DQEJARYAMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCJ9WRanG/fUvcfKiGl
    EL4aRLjGt537mZ28UU9/3eiJeJznNSOuNLnF+hmabAu7H0LT4K7EdqfF+XUZW/2j
    RKRYcvOUDGF9A7OjW7UfKk1In3+6QDCi7X34RE161jqoaJjrm/T18TOKcgkkhRzE
    apQnIDm0Ea/HVzX/PiSOGuertwIDAQABMAsGCSqGSIb3DQEBBQOBgQBzMJdAV4QP
    Awel8LzGx5uMOshezF/KfP67wJ93UW+N7zXY6AwPgoLj4Kjw+WtU684JL8Dtr9FX
    ozakE+8p06BpxegR4BR3FMHf6p+0jQxUEAkAyb/mVgm66TyghDGC6/YkiKoZptXQ
    98TwDIK/39WEB/V607As+KoYazQG8drorw==
    -----END CERTIFICATE-----
  cert: |-
    -----BEGIN CERTIFICATE-----
    MIIB9TCCAWACAQAwgbgxGTAXBgNVBAoMEFF1b1ZhZGlzIExpbWl0ZWQxHDAaBgNV
    BAsME0RvY3VtZW50IERlcGFydG1lbnQxOTA3BgNVBAMMMFdoeSBhcmUgeW91IGRl
    Y29kaW5nIG1lPyAgVGhpcyBpcyBvbmx5IGEgdGVzdCEhITERMA8GA1UEBwwISGFt
    aWx0b24xETAPBgNVBAgMCFBlbWJyb2tlMQswCQYDVQQGEwJCTTEPMA0GCSqGSIb3
    DQEJARYAMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCJ9WRanG/fUvcfKiGl
    EL4aRLjGt537mZ28UU9/3eiJeJznNSOuNLnF+hmabAu7H0LT4K7EdqfF+XUZW/2j
    RKRYcvOUDGF9A7OjW7UfKk1In3+6QDCi7X34RE161jqoaJjrm/T18TOKcgkkhRzE
    apQnIDm0Ea/HVzX/PiSOGuertwIDAQABMAsGCSqGSIb3DQEBBQOBgQBzMJdAV4QP
    Awel8LzGx5uMOshezF/KfP67wJ93UW+N7zXY6AwPgoLj4Kjw+WtU684JL8Dtr9FX
    ozakE+8p06BpxegR4BR3FMHf6p+0jQxUEAkAyb/mVgm66TyghDGC6/YkiKoZptXQ
    98TwDIK/39WEB/V607As+KoYazQG8drorw==
    -----END CERTIFICATE-----
  key: |-
    -----BEGIN EC PRIVATE KEY-----
    MHcCAQEEIC8CsJ/B115S+JtR1/l3ZQwKA3XdXt9zLqusF1VXc/KloAoGCCqGSM49
    AwEHoUQDQgAEpwUmRIZHFt8CdDHYm1ikScCScd2q6QVYXxJu+G3fQZ78ScGtN7fu
    KXMnQqVjXVRAr8qUY8yipVKuMCepnPXScQ==
    -----END EC PRIVATE KEY-----
```

jwksFile

JWT Header Key ID

If the JWT header contains a `kid` header, the middleware expects to find a JWK. If a JWK cannot be found, it returns a `401 Unauthorized` error.

jwksUrl

JWT Header Key ID

If the JWT header contains a `kid` header, the middleware expects to find a JWK. If a JWK cannot be found, it returns a `401 Unauthorized` error.

JWT Issuer Claim

If `jwksUrl` is set to a path and the `iss` property is missing in the JWT it's trying to verify, the middleware returns a `401 Unauthorized` error.

signingSecret

Storing secret values in Kubernetes secrets

When configuring the `signingSecret`, it is possible to reference a Kubernetes secret defined in the same namespace as the Middleware. The reference to a Kubernetes secret takes the form of a URN:

```
urn:k8s:secret:[name]:[valueKey]
```

{% include-markdown "includes/traefik-for-business-applications.md" %}

□□□ #2
□ □□□ □□□ 2026-10-31 14:03:22 UTC
□ □□□ □□□ 2026-10-31 14:03:23 UTC