JWT library free jwt user generation

Metadata	Value
Date	2021-07-20
Author	@mhanel, @aricart
Status	Approved
Tags	client, security

Context and Problem Statement

Users writing programs to issue user jwt, ask if this can be done in **languages other than go**. We assume that these programs stamp out a set number of identical user profiles.

With the addition of scoped signing keys, limits and permissions are **attached to the signing key instead of the user**. As a consequence, user jwt generating programs, do not need the data model the jwt library provides. The use of scoped signing keys is highly encouraged as a compromised key can only be used to issue certain user, NOT user with arbitrary permissions.

Thus, a simple utility function, added to nkey supporting client libraries, can replace the need for the go only jwt library, for most use cases.

> This can be a series of standalone repositories as well. Because this is a single and small utility function, adding it to suitable client libraries seems easiest for usage as it meets the users where they are (similar to nkey functions in some languages). Furthermore, this may not be needed for every client library. Initially we'd focus on languages widely used by corporate user where jwt generating programs are to be expected. Those would be: `go`, `java`, `csharp`

References

This is going to be a much simplified version of this c# example. Instead of the go jwt library with the full data model, `sprintf` is used with very few values. Any library implementing this will require basic `nkey` functions to `sign` and `generate` new nkeys. `sign` will exist in every library supporting nkeys. Generating new nkeys is a matter of base64url random bytes, base64url encoding and decorating them.

Design

The proposed function is supposed to only set values that are not covered by the scoped signing key. These values are `name`, `expiration`, `account id`, `tags`. The `user nkey` can be generated by the signer or a requestor can provide it so the signer is unaware of the private nkey portion. So that the user can generate nkeys when not provided, the functionality to generate a user nkey needs to be public as well.

Proposed name and function signature:

```
/**
 * signingKey, is a mandatory account nkey pair to sign the generated jwt.
 * accountId, is a mandatory public account nkey. Will return error when not set
or not account nkey.
 * publicUserKey, is a mandatory public user nkey. Will return error when not set
or not user nkey.
 * name, optional human readable name. When absent, default to publicUserKey.
 * expiration, optional but recommended duration, when the generated jwt needs to
expire. If not set, JWT will not expire.
 * tags, optional list of tags to be included in the JWT.
 *
 * Returns:
 * error, when issues arose.
 * string, resulting jwt.
 **/
IssueUserJWT(signingKey nkey, accountId string, publicUserKey string, name string,
expiration time.Duration, tags []string) (error, string)
```

Static header json:

```
{"typ":"JWT","alg":"ed25519-nkey"}
```

Commented body (//) json output that is subsequently turned into a jwt (treat not commented values as static):

```
{
 // expiration (when specified) time in unix seconds, derived from provided
duration.
 "exp": 1632077055,
 // issue time in unix seconds. Will always need to be set.
 "iat": 1626720255,
 // public nkey portion of signingKey
 "iss": "AACYICOAQMQ72EHT35R7LV6VFWMIVWFKWFE5P2JJ2TT674EO7DJTUHMM",
 // unique implementation dependent value
 // our jwt library uses base32 encoded sha256 hash of this json with "jti":""
 "jti": "FRQCL7TPAIL6KHKPPPJS2YOHTANKNHTPLTX7STGPTGYZVGTOH2LQ",
 // name provided or value of publicUserKey
 "name": "USER_NAME",
 "nats": {
  // value of accountId
  "issuer_account": "ADECCNBUEBWZ727OMBFSN7OMK2FPYRM52TJS25TFQWYS76NPOJBN3KU4",
  // tags, omit fully when not provided
  "tags": [
   "provided_tag1",
   "provided_tag2"
  ],
  "type": "user",
  "version": 2
 },
 // public nkey portion of signingKey
 "sub": "UD44C3VDAEYG527W3VPY353B3C6LIWJNW77GJED7MM5WIPGRUEVPHRZ5"
}
```

The `jti` can be a random value. If possibly, we should compute it the same way the jwt library does:

1. The body is first generated with `jti` set to `""`.
2. Then the `jti` value is computed as base32 encoded sha256 hash of the body.
3. Then the body is generated again, this time with the correct `jti`.

Both the static header json as well as the body are then base64url encoded. Please note that base64url is slightly different than base64. Padding characters `=` are removed and `+` is changed to `-` and `/` is changed to `_`. Compute the signature using signingKey over the following string (The dot is included as per jwt specification):

```
base64url-header + "." + base64url-body
```

Encode the signature with base64url, concatenate and return the resulting token:

```
base64url-header + "." + base64url-body + "." + base64url-signature
```

Consequences

Remove the need for the jwt library when programmatically generating user jwt using scoped signing keys.

Scoped Signing Key Setup for Development

These commands generate an account named ACC, generate and add a signing key to it, and promote the generated signing key to a scoped one with an arbitrary role name for easier referencing, issue the next commands:

```
> nsc add account -n ACC
> nsc edit account --name ACC --sk generate
> nsc edit signing-key --account ACC --sk <key output in previous command> --role
anyrolename <any limit/permissions option you want>
```

To add a user with values similar to the earlier json body and sign it with the scoped signing key and output the json, execute:

```
> nsc add user --account ACC --name USER_NAME --tag PROVIDED_TAG1 --tag
PROVIDED_TAG2  --expiry 2h --private-key anyrolename
> nsc describe user --account ACC --name USER_NAME --json
```

An invocation of the utility function, generating a similar user would look like this:

```
IssueUserJWT(accSigningKey,
"ADECCNBUEBWZ727OMBFSN7OMK2FPYRM52TJS25TFQWYS76NPOJBN3KU4",
"UD44C3VDAEYG527W3VPY353B3C6LIWJNW77GJED7MM5WIPGRUEVPHRZ5", "USER_NAME",
2*time.Hour, []string{"PROVIDED_TAG1", "PROVIDED_TAG2})
```