GNAP Meeting
IETF 115

draft-ietf-gnap-core-protocol-11
draft-ietf-gnap-resource-servers-02

November 10, 2022

Justin Richer • Fabien Imbault

MIND THE GNAP

Agenda

- Core draft update: changes since IETF114 (from -10 to -11)
  - Editorial Changes
  - Functional Changes
- RS draft update: no changes since IETF114 (-02)
- GNAP Futures

Differences since IETF114 (Core: -10 to -11)

[https://www.ietf.org/rfcdiff ?url2=draft-ietf-gnap-core-protocol-11 &url1=draft-ietf-gnap-core-protocol-10](https://www.ietf.org/rfcdiff?url2=draft-ietf-gnap-core-protocol-11&url1=draft-ietf-gnap-core-protocol-10)

23 (core) & 1 (RS) Merged Pull Requests

https://github.com/ietf-wg-gnap/gnap-core-protocol/pulls?q=is%3Aclosed+closed%3A2022-07-12..2022-10-24

https://github.com/ietf-wg-gnap/gnap-resource-servers/pulls?q=is%3Aclosed+closed%3A2022-07-12..2022-10-24

55 (core) closed issues

https://github.com/ietf-wg-gnap/gnap-core-protocol/issues
?q=is%3Aissue+is%3Aclosed+closed%3A2022-07-12..2022-10-24

No closed issues on the RS draft

Functional Changes Summary

- Key rotation of issued access tokens
- Cross-user authorization
- Consistent syntax (eg, start methods, access rights, key formats, key proof methods, etc.)
- Guidance for extensions and IANA registries
- All provided URIs are absolute
- Expanded error messages and syntax
- Expanded RS-first discovery to include referrer information

Editorial Changes Summary

- Extended security considerations
- Explanation on the use of class_id
- Update grant definition (the act of granting permission to a client instance.)
- Interoperability profiles (MTI considerations)
- Implementation status section
- Remove unused sections (eg, models)
- Clean up inline issues

JSON Schema

- Not in core document – on wiki
- Aims to simplify the understanding and verification of the GNAP model
- https://json-schema.org provides a rich set of tools
- Resource for implementors, available at https://github.com/ietf-wg-gnap/gnap-core-protocol/wiki/Implementation-guidelines#json-schema

Key Formats

- **Previously:** keys could be in all formats at once as long as it was the same key value
- **However:** if you can’t understand a key format, you can’t make sure it’s the same key in all formats
  - And the only reason to send multiple formats is if you don’t know what formats the receiver understands ahead of time
  - Client software is usually just going to be configured with a single key in a single format
- **Now:** only one key format per message
  - Have to choose between JWK, cert, etc

Key Rotation for Tokens

- Non-bearer tokens have keys bound to them
  - We want to have ways of binding new keys to these tokens
- Need a way to prove possession of old and new key simultaneously
  - Old key: right to change it
  - New key: proof you have it
- Each proof method defines its own way to do this
  - No way to change proof methods mid-stream (get a new token if you need to)
- Token management API can take parameters:
  - “key”: new key to bind to
  - “previous_key”: old bound key, for reference
  - If “key” is present then old and new key bindings apply

Key Rotation with HTTPSig

- HTTP Message Signatures allow multiple signatures within a single message
- Include the new key in the token rotation message
- Sign the message with the old key to prove you still have it
- Sign the old signature with the new key

```
Signature: sig1=("signature";key="sig2" ...);keyid= old_key,
           sig2=(...);keyid= new_key
```

```
{
  "proof": "httpsig",
  "key": { "jwk" ... } // new key included in message
  "previous_key": { "jwk" ... } // previous key included for reference
}
```

Key Rotation with JOSE

- Put the new key in the message
- Sign the outer message with the old key
- Take the signed object as the payload of a new JWS and sign that with the new key

```
Detached-JWS: eyj0... // message signed with old key then new key
```

```
{
  "proof": "jwsd",
  "key": { ... } // new key (signed with old key)
  "previous_key": { ... } // old key (for reference)
}
```

Key Rotation with MTLS

- Many MTLS systems will rely on certificate management systems (PKI), so rotation is out of scope for GNAP
- Could use ACME’s proposed client certificate extensions, but that is left to extension work

Cross-user Asynchronous Authorization

- Client instance states who it wants subject information about
  - New “sub_ids” field, mirrors use in other parts of protocol
- This can differ from who the client instance thinks is there right now (“user”)
  - Supporting advanced use cases like CIBA

```
"subject": {
  "sub_ids": [{"format": "email", "email": "customer@example.com"}]
},
"user": {
  "sub_ids": [{"format": "email", "email": "admin@example.net"}]
}
```

Syntax Alignment

- Several items defined by type: “object” or “string”
  - Access rights, key proof methods, interaction start methods
- Objects allow additional parameters
- All mappings need to be explicitly defined

Object:

```
{
  "proof": {
    "method": "httpsig",
     "alg": "ecdsa-p384-sha384",
     "content-digest-alg": "sha-256"
  }
}
```

String:

```
{
  "proof": "httpsig"
}
```

Extended security considerations

- Referencing BCP 107 / RFC4107 (Guidelines for Cryptographic Key Management)
- Considerations around key binding in addition to TLS, referencing I-D.ietf-uta-6125bis (service names in TLS)

Class_id

- Hint to the AS
- Pre-registration is possible
- Could be a hardcoded parameter (FoobarTV)
- Dynamic context (e.g. matches a field in certificate)

Interoperability profiles / Mandatory to implement

Web profile	**Secondary device**
Interaction Start Methods: redirect Interaction Finish Methods: redirect Interaction Hash Algorithms: sha3-512 Key Proofing Methods: httpsig Key Formats: jwks JOSE Signature Algorithm: PS256 Subject Identifier Formats: opaque Assertion Formats: id_token	Interaction Start Methods: user_code + user_code_uri Interaction Finish Methods: push Interaction Hash Algorithms: sha3-512 Key Proofing Methods: httpsig Key Formats: jwks JOSE Signature Algorithm: PS256 Subject Identifier Formats: opaque Assertion Formats: id_token

Implementation status (please add yours!)

- **GNAP Authorization Service in Rust** implementation by David Skyberg. https://github.com/dskyberg/gnap. Prototype implementation of AS and Client in Rust. MIT license.
- **Rafiki** from Interledger Foundation. https://github.com/interledger/rafiki Production implementation of AS in JavaScript. Apache 2.0 license.
- **Sample GNAP Client in PHP** implementation by Aaron Parecki. https://github.com/aaronpk/gnap-client-php Prototype implementation of web application client and CLI client in PHP, with common support library. CC0 license.
- **SUNET Auth Server** from SUNET. https://github.com/SUNET/sunet-auth-server Production implementation of AS in Python. BSD license.
- **XYZ** from Bespoke Engineering, implementation by Justin Richer. https://github.com/bspk/oauth.xyz-java. Advanced prototype implementation of AS, Client, and RS in Java, with common support library. Prototype implementation of SPA Client in pure JavaScript. Apache 2.0 license.

Current Open Issues on Core

- Key rotation: errors and discovery
  - Should be a simple mechanical fix
- Authentication: subject identifiers are scoped to AS
  - Previously discussed: subject identifiers cannot be global
  - Need to say this explicitly in requirements for clients

GNAP Futures

- Core to WGLC
  - Editors think it’s pretty much ready (could use an editorial cleanup pass)
- Is there energy to finish RS?