GNAP Meeting IETF 111

draft-ietf-gnap-core-protocol-06
draft-ietf-gnap-resource-servers-01

July 26, 2021

Justin Richer • Aaron Parecki • Fabien Imbault

Agenda

- Core draft update: changes since IETF111 (from -04 to -06)
    - Editorial Changes
    - Functional Changes
- RS draft update: changes since IETF111 (from `null` to -01)
- Topic: Mixup Attack
- Topic: Removed Features
- Draft roadmap: overview of next big topics
    - Topic: Key rotation
- Implementations

Differences since IETF111 (-04 to -06)

https://www.ietf.org/rfcdiff
?url2=draft-ietf-gnap-core-protocol-06
&url1=draft-ietf-gnap-core-protocol-04

https://www.ietf.org/archive/id/
draft-ietf-gnap-resource-servers-01.html

36 (core) & 10 (RS) Merged Pull Requests

https://github.com/ietf-wg-gnap/gnap-core-protocol/pulls?q=is%3Apr+is%3Aclosed+merged%3A2021-02-23..2021-07-12

https://github.com/ietf-wg-gnap/gnap-resource-servers/pulls?q=is%3Apr+is%3Aclosed+merged%3A2021-02-23..2021-07-12

Functional Changes

- Updated discovery mechanism
    - #183, #194, #269
- Subject identifiers
    - Syntax changes: #184, #228, #229
    - Subject identifier format: #220
    - Add DID examples: #274
- Cryptography/Signing
    - DPoP: #195
    - Normalize htu claim: #202
    - Editorial changes for JWS: #207
    - Access tokens: #208, #209
    - Type parameter for JWS methods: #226
    - Describing keys: #232
    - Crypto requirements: #250

Functional Changes

- Cache-Control header: #199
- Extracting the RS communication into its own spec
  - Initial extraction: #246
  - RS-first discovery: #261
- Authorization interaction
  - Describing options for interacting with the user: #242
- Privileges field: #259
- Added new parameter for mixup: #268
- Removing features
  - “Extension capabilities” and “Existing grant”: #270
  - DPoP: #271
  - OAuth PoP: #272

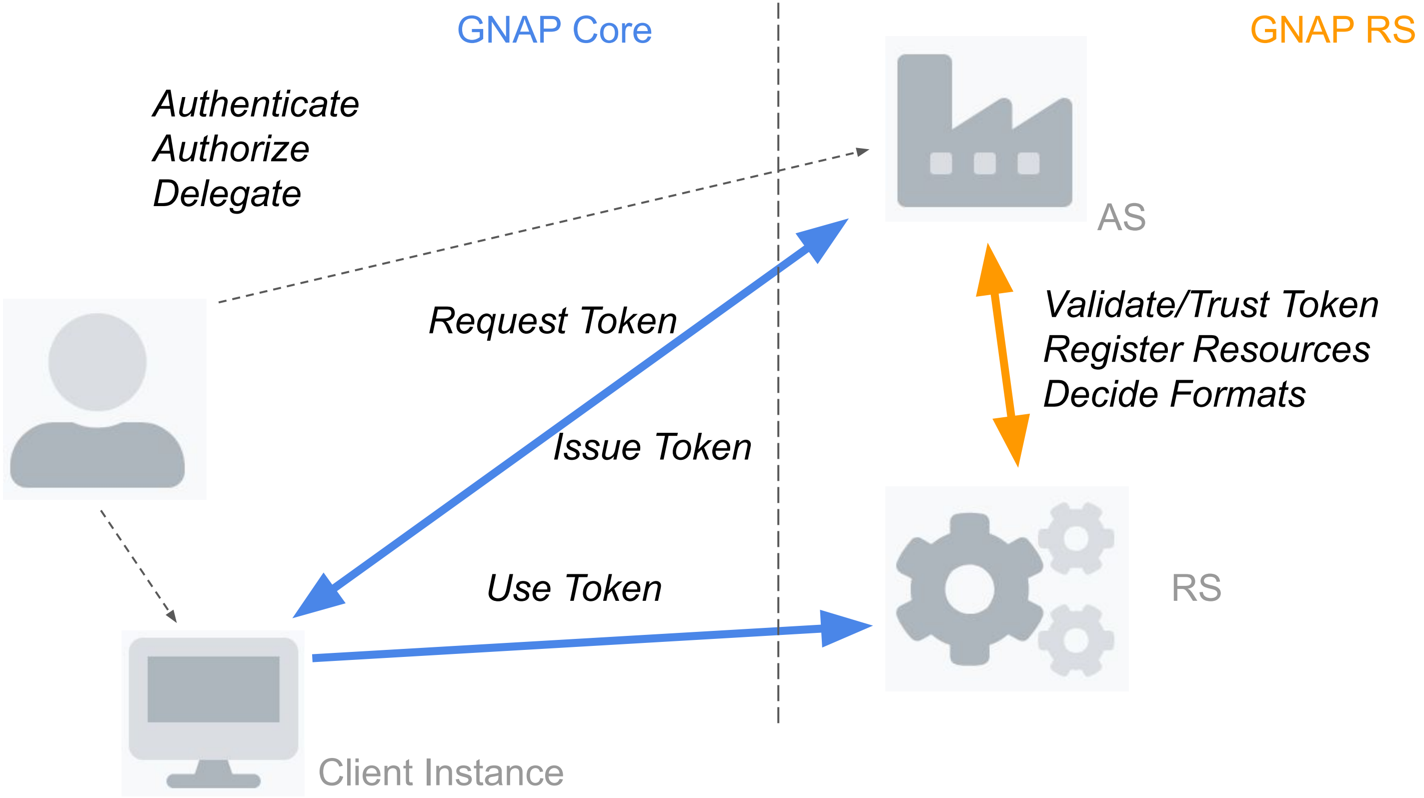
Editorial Changes

- Fixing some examples: #200
- Fixing typos: #204, #251
- Updating diagrams
  - #211, #245
- Protocol rationale:
  - #247, #245
  - Added diagram: #267
- Updates flags for consistency: #273
- Also done:
  - Editorconfig for document consistency
  - Some post -06 typo/format fixes (thank you!)

Document Structure Changes

- Extracted RS-facing components to draft-ietf-gnap-resource-servers
  - Expanded text discussion and cleaned up language for -01
- Things that affect RS interoperability but not clients
  - Resource set registration/introduction
  - Token introspection
  - Token formats
  - RS discovery of AS attributes

AS and RS Relationship

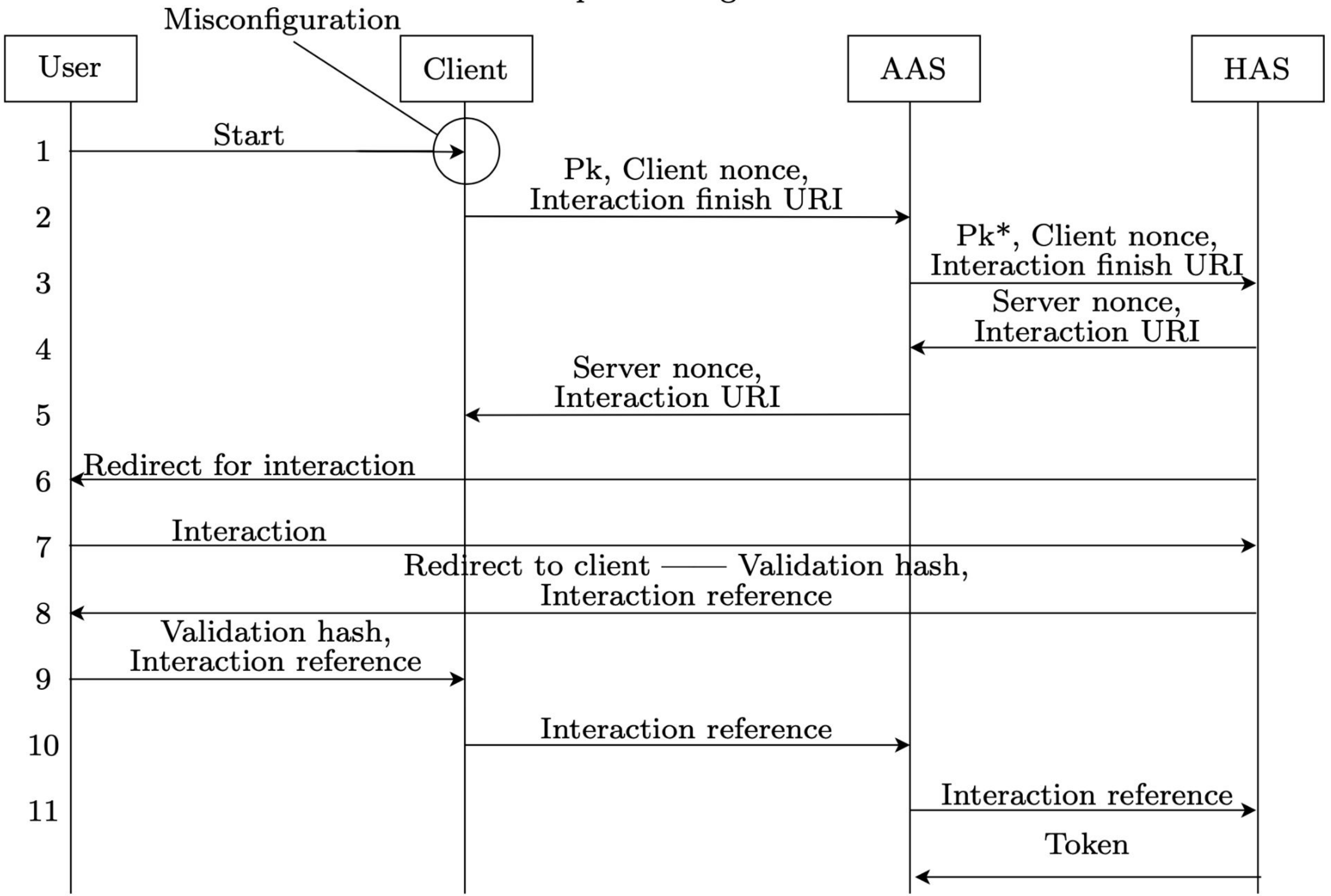


GNAP Mix-Up Attack

GNAP Mix-Up Attack

- Related to the OAuth 2 “Mix Up” attack
- Mitigation already proposed (extend interaction hash)
- How it works:
  - Attacker gets uncompromised client (UC) to talk to attacker AS (AAS)
  - AAS acts as a different client instance to home AS (HAS)
  - AAS proxies UC’s request to HAS to start transaction and kick off interaction
  - User interacts with HAS and approves AAS
  - UC gives reference back to AAS, AAS gets token
- How it’s different from OAuth 2
  - Client requests are bound to keys instead of bare secrets: no impersonation on the wire
  - Access token is (normally) bound to a key

AS mix-up attack aginst GNAP



Attack Steps

1. UC is a client of AAS, and might also be a client of HAS. User wants to authorize at HAS but tells UC to use AAS.
2. UC starts a request at AAS, signed with UC's key. AAS is imitating HAS.
3. AAS forwards UC's request parameters (Client nonce, interaction finish URI) to HAS, but signed with AAS's key.
4. HAS responds with an interaction start URL and server nonce to AAS
5. AAS forwards the interaction start URL and server nonce to UC
6. (Note) HAS is functionally telling the user to show up and interact, but doesn't realize that the request is being proxied by AAS from UC in this way.
7. UC launches interaction start url, which is a function of HAS
8. HAS returns the verification hash and interaction reference to UC
9. UC validates the hash (which is correct) and sends the interaction reference to AAS
10. AAS forwards the interaction reference to HAS
11. AAS receives an access token for calling an RS protected by HAS. The client receives no access token.

Mitigation (implemented in -06)

- Add the grant endpoint URL to the interaction hash calculation
  - Known to client instance
  - Known to HAS (and its interaction elements)
  - Known to AAS but can’t be modified or substituted
- Against this attack:
  - HAS uses its own URL to generate hash
  - UC uses the AAS URL to generate validation hash
  - UC hash validation fails and attack stops before interaction reference is presented to AAS
- Similar to OAuth 2 “iss” return parameter, but cryptographically bound

Mitigation discussion

- Redirect-based protocols are inherently phishable
- Methods without interaction “finish” susceptible to similar phishing attacks during polling periods
- Attack is made easier by dynamic clients but possible even with static clients
  - AAS impersonates UC to the user
  - Attacker gets UC to talk to AAS in the first place but convinces user that they’re using HAS

Removed Features

Signature Methods

- Kept:
  - HTTP Message Signatures
  - MTLS
- Dropped:
  - OAuth PoP (deprecated/expired)
  - DPoP (not a good fit for general signatures)
- Open for discussion:
  - Attached JWS
  - Detached JWS

Capabilities

- Was part of a proposed extension-discovery mechanism
- No proposed extensions used it
- Removed pending a use case to drive it
- Need to have clearer text for extensions overall

Existing Grant

- Ability to create a new request based on an existing one
  - Without updating the referenced grant
  - Awkward use of access tokens to identify requests
- With API-based grant update, not as needed now
- Could be added back as a more fully-thought-out extension

Alternative client instance identifier

- Ability to reference a client instance identifier inside of a client object alongside other fields
- No driving use case required this
- Could be brought back in a more formal client instance management API
    - Part of key rotation discussion

Discussion Items

Draft Roadmap

- Key rotation
    - Mailing list discussion: need to define a mechanism as part of GNAP, but also enable any existing mechanism (key reference)
- Trust relationships
- Security and privacy considerations
- Extension discussion
    - IANA Registries

Key Rotation

- For client instances
  - Client management API?
- For ongoing grants
  - Grant update API?
  - Does this also rotate key for client/token?
- For access tokens
  - Part of token management API?
  - Could client instance use different keys for different tokens?

Implementation

Implementation status

- Java implementation updated to latest draft
    - Python, PHP, and Rust in the works
- Dependency implementations:
    - HTTP Message Signatures implementations (Java, Python, Go)
    - SECEVENT identifier implementations (Java, Python, JS, Rust)
- Editors will add an implementation status section to core draft
- Major churn has died down
    - Syntax and details are still being bikeshedded

Open Discussion