IETF 82 - OAuth Bearer Token Comments

Julian Reschke, greenbytes

Background: How We Got here

- draft-ietf-oauth-v2-bearer is a deliverable of the OAuth WG (Security Area)
- defines an HTTP Authentication Scheme; the HTTPbis WG (Apps Area) is chartered to revise HTTP, including the Authentication Framework from RFC 2616 and 2617
- there was some *early* back-and-forth between the Working Groups (mainly thanks to James Manger)
- as a result, draft-ietf-oauth-v2-bearer now normatively refers to draft-ietf-httpbis-p7-auth (and this is good)
- the following slides illustrate remaining problems that I see from an Appsarea and HTTPbis point-of-view (but I'm not speaking in any official function, nor have I one, except being one of the editors of HTTPbis)

Background: draft-ietf-httpbis-p7-auth vs RFC 2617

- draft-ietf-httpbis-p7-auth contains the *framework*, but not the definitions of "Basic" and "Digest"
- lots of clarifications and bug fixes (such as Basic not being compatible with the RFC 2617 ABNF for credentials)
- adds IANA registry for authentication schemes, and provides guidelines
- please review, it's only ~15 pages excluding boilerplate and change logs

#1: Parameter Syntax (1/2)

see Section 3

- the syntax of the WWW-Authenticate and Authorization header fields is defined by HTTP, not by individual schemes definitions
- this is really important, as WWW-Authenticate allows sending multiple challenges in a single header field instance (and this will be needed if you want to roll out the Bearer scheme on existing web sites with a fallback to other schemes)

example from HTTPbis P7, Section 4.4:

```
WWW-Authenticate: Newauth realm="apps", type=1,
                  title="Login to \"apps\"", Basic realm="simple"
```

#1: Parameter Syntax (2/2)

- recipients **MUST** parse the header field using a generic parser capable of processing parameters using both "token" and "quoted-string" syntax
- recipients that do this usually do not care what notation a parameter was in (nor should they); in practice, the parser they use may not even tell them
- observation: test results for how popular User Agents parse "realm" (which the spec requires to be a quoted-string): http://greenbytes.de/tech/tc/httpauth/#simplebasictok - all of them also accept the token format (HTTPbis issue #314)
- conclusion: it's best to allow both notations for all parameters in newly defined schemes; many recipients will accept them no matter what the spec says, thus causing potential interop problems

#2: Use of application/x-www-form-urlencoded media type (1/2)

see Section 2.2 -- *how do you submit an extension parameter containing a non-ASCII character?*

- HTML 4.01 defines this media type for the purpose of form submissions, in which case the character encoding to be used depends on the character encoding of the document containing the form, or properties on the <form> element (accept-charset, I believe)
- the definition as cited from Section 17.13.3.4 of HTML 4.01 leaves character encoding undefined, and thus isn't suitable for use without out-of-band information about the character set
- note that the definition in Section 4.10.22.5 of HTML5 is more complete
- as currently defined in the Bearer spec, the encoding is underspecified for non-ASCII characters (and even for ASCII characters, in case of nitpicking...)

#2: Use of application/x-www-form-urlencoded media type (2/2)

- hint: enter something into a Google search form and see the "ie=" parameter in the submitted request

choices:

1. declare that the encoding is always US-ASCII, both for predefined parameters or extension parameters (making I18N of extension parameters impossible)
2. declare that the encoding is always UTF-8, both for predefined parameters or extension parameters
3. declare that the encoding for predefined parameters is either US-ASCII or UTF-8, and that extension parameters will need to define it themselves
4. declare some other mechanism specifiying the charset (like "ie=")

#3: encoding in URI query parameters

see Section 2.3

this essentially is the same problem as for the form encoding