Web Server Core Design (Part 2)

BY EVAN OLDS
CPT S 422

HTTP Request Headers

- The HTTP headers in a request give us various information about the request
  - Information about the browser or other agent that is making the request
  - Information about what types of data can be sent back
  - Size, in bytes, of the request body
  - (many more things..)
- All headers are optional according to the actual standard, and as we saw in a previous lecture, but a web app may choose to accept or reject requests if some are missing

HTTP Response Headers

- Same idea as the request headers, but these are sent in the response
- Some headers may be required by the client in order to proceed with a transaction
  - As an example, some response headers may dictate whether or not cross-origin resource sharing

Header Fields Format

- From RFC7230 section 3.2:

Each header field consists of a case-insensitive field name followed by a colon (":"), optional leading whitespace, the field value, and optional trailing whitespace.

Header Field Length Limits

- From RFC 7230 section 3.2.5 (Field Limits):

HTTP does not place a predefined limit on the length of each header field or on the length of the header section as a whole, as described in Section 2.5. Various ad hoc limitations on individual header field length are found in practice, often depending on the specific field semantics.

A server that receives a request header field, or set of fields, larger than it wishes to process MUST respond with an appropriate 4xx (Client Error) status code. Ignoring such header fields would increase the server's vulnerability to request smuggling attacks (Section 9.5).

A client MAY discard or truncate received header fields that are larger than the client wishes to process if the field semantics are such that the dropped value(s) can be safely ignored without changing the message framing or response semantics.

HTTP Headers

- There are a lot of official (and unofficial) headers out there
- We care about a select few that are of relevance to the task at hand
- Recall problem from previous lecture: Request object body stream
- The Content-Length header is of interest to us as we finish implementing a solution to that problem

Content-Length Header

- Base-10 number as a string that indicates size, in bytes, of the body of the request
- Example:
  - Content-Length: 7146
- Can give us the size of the body so we don't have to "guess" or just keep accepting more and more data until the socket connection closes

Content-Length Header

- Watch out for attacks / implementation quirks: What if provided content length was:
  - (attack) Intentionally longer than the actual body of the request
  - (attack) Intentionally shorter than the body
  - (implementation) A value that can’t be parsed as an integer because it goes beyond int32.MaxValue
  - (attack) Intentionally not parsable as an integer type because it is gibberish or not “fully” a number
    - “ABCD”
    - “256Hello”
    - “$&%#*Q*#&”

Content-Length Header

- With information about the length of the body, we can create a Stream to put in the request object
  - Recall that this request object will be passed to a handler
  - The handler doesn’t want to have to do much other than analyze already read parts of the request or read from the body stream
  - NetworkStream doesn’t support querying length, but we can have a custom stream class that represents the body and has a length that came from the Content-Length header
  - (discuss on whiteboard in class)