aio-libs / **aiohttp** Public

Code Issues 184 Pull requests 90 Discussions Actions Security an

Commit 53e2e6f

3 people authored 4 days ago · ✕ 5 / 37 · Verified

```
[PR #12302/2dc02ee0 backport][3.13] Skip duplicate singleton header check in lax
mode (#12303)

Co-authored-by: J. Nick Koston <nick@koston.org>
Co-authored-by: pre-commit-ci[bot] <66853113+pre-commit-ci[bot]@users.noreply.github.com>
Fixes home-assistant/core#166956
Fixes getmoto/moto#9930
Fixes #12301
Fixes catalyst-cooperative/pudl-archiver#1059
```

master (#12303) · v3.13.5

1 parent 9f7c7ab commit 53e2e6f

4 files changed +131 -36 lines changed

Top

Filter files…

- CHANGES
  - 12302.bugfix.rst
- aiohttp
  - _http_parser.pyx
  - http_parser.py
- tests
  - test_http_parser.py

4 files changed +131 -36 lines changed

Search within code

CHANGES/12302.bugfix.rst

```
@@ -0,0 +1,3 @@
```

```
+ Skipped the duplicate singleton header check in lax mode (the default for response
+ parsing). In strict mode (request parsing, or ``-X dev``), all RFC 9110 singletons
+ are still enforced -- by :user:`bdraco`.
```

aiohttp/_http_parser.pyx

```
@@ -71,8 +71,11 @@ cdef object StreamReader = _StreamReader
 cdef object DeflateBuffer = _DeflateBuffer
 cdef bytes EMPTY_BYTES = b""

-# https://www.rfc-editor.org/rfc/rfc9110.html#section-5.5-6
-cdef tuple SINGLETON_HEADERS = (
+# RFC 9110 singleton headers — duplicates are rejected in strict mode.
+# In lax mode (response parser default), the check is skipped entirely
+# since real-world servers (e.g. Google APIs, Werkzeug) commonly send
+# duplicate headers like Content-Type or Server.
+cdef frozenset SINGLETON_HEADERS = frozenset({
     hdrs.CONTENT_LENGTH,
     hdrs.CONTENT_LOCATION,
     hdrs.CONTENT_RANGE,
@@ -83,7 +86,7 @@ cdef tuple SINGLETON_HEADERS = (
     hdrs.SERVER,
     hdrs.TRANSFER_ENCODING,
     hdrs.USER_AGENT,
-)
+})

 cdef inline object extend(object buf, const char* at, size_t length):
     cdef Py_ssize_t s
@@ -304,13 +307,15 @@ cdef class HttpParser:
         size_t _max_headers
         bint _response_with_body
         bint _read_until_eof
+        bint _lax

     bint    _started
     object  _url
     bytearray   _buf
```

```
311  315            str     _path
312  316            str     _reason
313  317            list    _headers
     318  +         set     _seen_singletons
314  319            list    _raw_headers
315  320            bint    _upgraded
316  321            list    _messages
     @@ -377,6 +382,8 @@ cdef class HttpParser:
377  382            self._upgraded = False
378  383            self._auto_decompress = auto_decompress
379  384            self._content_encoding = None
     385  +         self._lax = False
     386  +         self._seen_singletons = set()
380  387
381  388            self._csettings.on_url = cb_on_url
382  389            self._csettings.on_status = cb_on_status
     @@ -405,6 +412,10 @@ cdef class HttpParser:
405  412                if "\x00" in value:
406  413                    raise InvalidHeader(self._raw_value)
407  414
     415  +             if not self._lax and name in SINGLETON_HEADERS:
     416  +                 if name in self._seen_singletons:
     417  +                     raise BadHttpMessage(f"Duplicate '{name}' header found.")
     418  +                 self._seen_singletons.add(name)
408  419                self._headers.append((name, value))
409  420                if len(self._headers) > self._max_headers:
410  421                    raise BadHttpMessage("Too many headers received")
     @@ -444,14 +455,6 @@ cdef class HttpParser:
444  455            raw_headers = tuple(self._raw_headers)
445  456            headers = CIMultiDictProxy(CIMultiDict(self._headers))
446  457
447       -         # https://www.rfc-editor.org/rfc/rfc9110.html#name-collected-abnf
448       -         bad_hdr = next(
449       -             (h for h in SINGLETON_HEADERS if len(headers.getall(h, ())) > 1),
450       -             None,
451       -         )
452       -         if bad_hdr is not None:
```

```
453       -             raise BadHttpMessage(f"Duplicate '{bad_hdr}' header found.")
454       -
455   458             if self._cparser.type == cparser.HTTP_REQUEST:
456   459                 h_upg = headers.get("upgrade", "")
457   460                 allowed = upgrade and h_upg.isascii() and h_upg.lower() in
          ALLOWED_UPGRADES
          @@ -689,6 +692,7 @@ cdef class HttpResponseParser(HttpParser):
689   692                 cparser.llhttp_set_lenient_headers(self._cparser, 1)
690   693                 cparser.llhttp_set_lenient_optional_cr_before_lf(self._cparser, 1)
691   694                 cparser.llhttp_set_lenient_spaces_after_chunk_size(self._cparser,
          1)
          695   +             self._lax = True
692   696
693   697         cdef object _on_status_complete(self):
694   698             if self._buf:
          @@ -702,6 +706,7 @@ cdef int cb_on_message_begin(cparser.llhttp_t* parser)
          except -1:
702   706
703   707         pyparser._started = True
704   708         pyparser._headers = []
          709   +     pyparser._seen_singletons = set()
705   710         pyparser._raw_headers = []
706   711         PyByteArray_Resize(pyparser._buf, 0)
707   712         pyparser._path = None
```

aiohttp/http_parser.py

```
          @@ -89,6 +89,26 @@
89    89    DIGITS: Final[Pattern[str]] = re.compile(r"\d+", re.ASCII)
90    90    HEXDIGITS: Final[Pattern[bytes]] = re.compile(rb"[0-9a-fA-F]+")
91    91
          92    + # RFC 9110 singleton headers — duplicates are rejected in strict mode.
          93    + # In lax mode (response parser default), the check is skipped entirely
          94    + # since real-world servers (e.g. Google APIs, Werkzeug) commonly send
          95    + # duplicate headers like Content-Type or Server.
          96    + # Lowercased for case-insensitive matching against wire names.
          97    + SINGLETON_HEADERS: Final[frozenset[str]] = frozenset(
          98    +     {
          99    +         "content-length",
```

```
      100 +         "content-location",
      101 +         "content-range",
      102 +         "content-type",
      103 +         "etag",
      104 +         "host",
      105 +         "max-forwards",
      106 +         "server",
      107 +         "transfer-encoding",
      108 +         "user-agent",
      109 +     }
      110 + )
      111 +
 92   112
 93   113   class RawRequestMessage(NamedTuple):
 94   114       method: str
          @@ -218,6 +238,8 @@ def parse_headers(
218   238                elif _FIELD_VALUE_FORBIDDEN_CTL_RE.search(value):
219   239                    raise InvalidHeader(bvalue)
220   240
      241 +             if not self._lax and name in headers and name.lower() in
          SINGLETON_HEADERS:
      242 +                 raise BadHttpMessage(f"Duplicate '{name}' header found.")
221   243                headers.add(name, value)
222   244                raw_headers.append((bname, bvalue))
223   245
          @@ -531,24 +553,6 @@ def parse_headers(
531   553            upgrade = False
532   554            chunked = False
533   555
534       -         # https://www.rfc-editor.org/rfc/rfc9110.html#section-5.5-6
535       -         # https://www.rfc-editor.org/rfc/rfc9110.html#name-collected-abnf
536       -         singletons = (
537       -             hdrs.CONTENT_LENGTH,
538       -             hdrs.CONTENT_LOCATION,
539       -             hdrs.CONTENT_RANGE,
540       -             hdrs.CONTENT_TYPE,
541       -             hdrs.ETAG,
542       -             hdrs.HOST,
```

```
543         -             hdrs.MAX_FORWARDS,
544         -             hdrs.SERVER,
545         -             hdrs.TRANSFER_ENCODING,
546         -             hdrs.USER_AGENT,
547         -         )
548         -         bad_hdr = next((h for h in singletons if len(headers.getall(h, ())) >
              1), None)
549         -         if bad_hdr is not None:
550         -             raise BadHttpMessage(f"Duplicate '{bad_hdr}' header found.")
551         -
552   556           # keep-alive and protocol switching
553   557           # RFC 9110 section 7.6.1 defines Connection as a comma-separated list.
554   558           conn_values = headers.getall(hdrs.CONNECTION, ())
```

tests/test_http_parser.py

```
      @@ -269,32 +269,76 @@ def test_content_length_transfer_encoding(parser: Any)
      -> None:
269   269       "hdr",
270   270       (
271   271           "Content-Length",
      272 +         "Host",
      273 +         "Transfer-Encoding",
      274 +     ),
      275 + )
      276 + def test_duplicate_singleton_header_rejected(
      277 +     parser: HttpRequestParser, hdr: str
      278 + ) -> None:
      279 +     val1, val2 = ("1", "2") if hdr == "Content-Length" else ("value1",
              "value2")
      280 +     text = (
      281 +         f"GET /test HTTP/1.1\r\n"
      282 +         f"Host: example.com\r\n"
      283 +         f"{hdr}: {val1}\r\n"
      284 +         f"{hdr}: {val2}\r\n"
      285 +         "\r\n"
      286 +     ).encode()
      287 +     with pytest.raises(http_exceptions.BadHttpMessage, match="Duplicate"):
      288 +         parser.feed_data(text)
      289 +
```

```
                +
                + @pytest.mark.parametrize(
                +     "hdr",
                +     (
272   294           "Content-Location",
273   295           "Content-Range",
274   296           "Content-Type",
275   297           "ETag",
276         -         "Host",
277   298           "Max-Forwards",
278   299           "Server",
279         -         "Transfer-Encoding",
280   300           "User-Agent",
281   301       ),
282   302   )
283         - def test_duplicate_singleton_header_rejected(
                + def test_duplicate_non_security_singleton_header_rejected_strict(
284   304       parser: HttpRequestParser, hdr: str
285   305   ) -> None:
286         -     val1, val2 = ("1", "2") if hdr == "Content-Length" else ("value1",
              "value2")
                +     """Non-security singletons are rejected in strict mode (requests)."""
287   307       text = (
288   308           f"GET /test HTTP/1.1\r\n"
289   309           f"Host: example.com\r\n"
290         -         f"{hdr}: {val1}\r\n"
291         -         f"{hdr}: {val2}\r\n"
292         -         f"\r\n"
                +         f"{hdr}: value1\r\n"
                +         f"{hdr}: value2\r\n"
                +         "\r\n"
293   313       ).encode()
294   314       with pytest.raises(http_exceptions.BadHttpMessage, match="Duplicate"):
295   315           parser.feed_data(text)
296   316
297   317
                + @pytest.mark.parametrize(
                +     "hdr",
                +     (
                +         # Content-Length is excluded because llhttp rejects duplicates
```

```
322 + # at the C level before our singleton check runs.
323 + "Content-Location",
324 + "Content-Range",
325 + "Content-Type",
326 + "ETag",
327 + "Max-Forwards",
328 + "Server",
329 + "Transfer-Encoding",
330 + "User-Agent",
331 + ),
332 + )
333 + def test_duplicate_singleton_header_accepted_in_lax_mode(
334 +     response: HttpResponseParser, hdr: str
335 + ) -> None:
336 +     """All singleton duplicates are accepted in lax mode (response parser
      default)."""
337 +     text = (f"HTTP/1.1 200 OK\r\n{hdr}: value1\r\n{hdr}:
      value2\r\n\r\n").encode()
338 +     messages, upgrade, tail = response.feed_data(text)
339 +     assert len(messages) == 1
340 +
341 +
298 342   def test_duplicate_host_header_rejected(parser: HttpRequestParser) -> None:
299 343       text = (
300 344           b"GET /admin HTTP/1.1\r\n"
      @@ -306,6 +350,45 @@ def test_duplicate_host_header_rejected(parser:
      HttpRequestParser) -> None:
306 350           parser.feed_data(text)
307 351
308 352
353 + @pytest.mark.parametrize(
354 +     ("hdr1", "hdr2"),
355 +     (
356 +         ("content-length", "Content-Length"),
357 +         ("Content-Length", "content-length"),
358 +         ("transfer-encoding", "Transfer-Encoding"),
359 +         ("Transfer-Encoding", "transfer-encoding"),
360 +     ),
361 + )
362 + def test_duplicate_singleton_header_different_casing_rejected(
```

```
363 +     parser: HttpRequestParser, hdr1: str, hdr2: str
364 + ) -> None:
365 +     """Singleton check must be case-insensitive per RFC 9110."""
366 +     val1, val2 = ("1", "2") if "content-length" in hdr1.lower() else ("v1",
      "v2")
367 +     text = (
368 +         f"GET /test HTTP/1.1\r\n"
369 +         f"Host: example.com\r\n"
370 +         f"{hdr1}: {val1}\r\n"
371 +         f"{hdr2}: {val2}\r\n"
372 +         "\r\n"
373 +     ).encode()
374 +     with pytest.raises(http_exceptions.BadHttpMessage, match="Duplicate"):
375 +         parser.feed_data(text)
376 +
377 +
378 + def test_duplicate_host_header_different_casing_rejected(
379 +     parser: HttpRequestParser,
380 + ) -> None:
381 +     """Duplicate Host with different casing must also be rejected."""
382 +     text = (
383 +         b"GET /test HTTP/1.1\r\n"
384 +         b"host: evil.example\r\n"
385 +         b"Host: good.example\r\n"
386 +         b"\r\n"
387 +     )
388 +     with pytest.raises(http_exceptions.BadHttpMessage, match="Duplicate"):
389 +         parser.feed_data(text)
390 +
391 +
309 392   def test_bad_chunked(parser: HttpRequestParser) -> None:
310 393       """Test that invalid chunked encoding doesn't allow content-length to be
      used."""
311 394       text = (
```

Comments 0

Please sign in to comment.