⚠ This commit does not belong to any branch on this repository, and may belong to a fork outside of the repository.

Commit ff4e5c2

4 people authored on Feb 19, 2025 Partially verified

```
[3.9] gh-105704: Disallow square brackets ([ and ]) in domain names for parsed
URLs (GH-129418) (#129530)

(cherry picked from commit d89a5f6)

Co-authored-by: Seth Michael Larson <seth@python.org>
Co-authored-by: Peter Bierma <zintensitydev@gmail.com>
Co-authored-by: Łukasz Langa <lukasz@langa.pl>
```

· v3.9.25 ••• 3.9

1 parent f116a9c commit ff4e5c2

3 files changed +58 -3

Top

Filter files…

- Lib
  - test
    - test_urlparse.py
  - urllib
    - parse.py
- Misc/NEWS.d/next/Security
  - 2025-01-28-14-08-03.gh-issue-105704.EnhHxu.rst

Search within code

Lib/test/test_urlparse.py

```
@@ -1146,16 +1146,51 @@ def test_invalid_bracketed_hosts(self):
1146 1146          self.assertRaises(ValueError, urllib.parse.urlsplit,
'Scheme://user@[0439:23af::2309::fae7:1234]/Path?Query')
1147 1147          self.assertRaises(ValueError, urllib.parse.urlsplit,
'Scheme://user@[0439:23af:2309::fae7:1234:2342:438e:192.0.2.146]/Path?Query')
1148 1148          self.assertRaises(ValueError, urllib.parse.urlsplit,
'Scheme://user@]v6a.ip[/Path')
     1149 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[v6a.ip]')
     1150 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[v6a.ip].suffix')
     1151 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[v6a.ip]/')
     1152 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[v6a.ip].suffix/')
     1153 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[v6a.ip]?')
     1154 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[v6a.ip].suffix?')
     1155 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]')
     1156 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix')
     1157 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]/')
     1158 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix/')
     1159 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]?')
     1160 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix?')
     1161 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]:a')
     1162 +          self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix:a')
```

```
1163  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://prefix.[::1]:a1')
1164  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://[::1].suffix:a1')
1165  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://prefix.[::1]:1a')
1166  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://[::1].suffix:1a')
1167  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://prefix.[::1]:')
1168  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://[::1].suffix:/')
1169  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://prefix.[::1]:?')
1170  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://user@prefix.[v6a.ip]')
1171  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://user@[v6a.ip].suffix')
1172  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://[v6a.ip')
1173  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://v6a.ip]')
1174  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://]v6a.ip[')
1175  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://]v6a.ip')
1176  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://v6a.ip[')
1177  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://prefix.[v6a.ip')
1178  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://v6a.ip].suffix')
1179  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://prefix]v6a.ip[suffix')
1180  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://prefix]v6a.ip')
1181  +         self.assertRaises(ValueError, urllib.parse.urlsplit,
      'scheme://v6a.ip[suffix')
1149  1182
1150  1183      def test_splitting_bracketed_hosts(self):
```

```
1151      -         p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]/path?query')
      1184  +         p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]:1234/path?query')
1152  1185            self.assertEqual(p1.hostname, 'v6a.ip')
1153  1186            self.assertEqual(p1.username, 'user')
1154  1187            self.assertEqual(p1.path, '/path')
      1188  +         self.assertEqual(p1.port, 1234)
1155  1189            p2 =
                  urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7%test]/path?query')
1156  1190            self.assertEqual(p2.hostname, '0439:23af:2309::fae7%test')
1157  1191            self.assertEqual(p2.username, 'user')
1158  1192            self.assertEqual(p2.path, '/path')
      1193  +         self.assertIs(p2.port, None)
1159  1194            p3 =
                  urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7:1234:192.0.2.146%t
                  est]/path?query')
1160  1195            self.assertEqual(p3.hostname,
                  '0439:23af:2309::fae7:1234:192.0.2.146%test')
1161  1196            self.assertEqual(p3.username, 'user')
```

Lib/urllib/parse.py

```
      @@ -443,6 +443,23 @@ def _checknetloc(netloc):
443   443               raise ValueError("netloc '" + netloc + "' contains invalid " +
444   444                                "characters under NFKC normalization")
445   445
      446 + def _check_bracketed_netloc(netloc):
      447 +     # Note that this function must mirror the splitting
      448 +     # done in NetlocResultMixins._hostinfo().
      449 +     hostname_and_port = netloc.rpartition('@')[2]
      450 +     before_bracket, have_open_br, bracketed = hostname_and_port.partition('[')
      451 +     if have_open_br:
      452 +         # No data is allowed before a bracket.
      453 +         if before_bracket:
      454 +             raise ValueError("Invalid IPv6 URL")
      455 +         hostname, _, port = bracketed.partition(']')
      456 +         # No data is allowed after the bracket but before the port delimiter.
      457 +         if port and not port.startswith(":"):
      458 +             raise ValueError("Invalid IPv6 URL")
      459 +     else:
      460 +         hostname, _, port = hostname_and_port.partition(':')
```

```
      461 +     _check_bracketed_host(hostname)
      462 +
446   463     # Valid bracketed hosts are defined in
447   464     # https://www.rfc-editor.org/rfc/rfc3986#page-49 and
                https://url.spec.whatwg.org/
448   465     def _check_bracketed_host(hostname):
          ⇣⇡  @@ -506,8 +523,7 @@ def urlsplit(url, scheme='', allow_fragments=True):
506   523                     (']' in netloc and '[' not in netloc)):
507   524                 raise ValueError("Invalid IPv6 URL")
508   525             if '[' in netloc and ']' in netloc:
509         -               bracketed_host = netloc.partition('[')[2].partition(']')[0]
510         -               _check_bracketed_host(bracketed_host)
      526 +               _check_bracketed_netloc(netloc)
511   527         if allow_fragments and '#' in url:
512   528             url, fragment = url.split('#', 1)
513   529         if '?' in url:
          ⇣
```

...5-01-28-14-08-03.gh-issue-105704.EnhHxu.rst

```
      ••• @@ -0,0 +1,4 @@
      1   + When using :func:`urllib.parse.urlsplit` and :func:`urllib.parse.urlparse` host
      2   + parsing would not reject domain names containing square brackets (``[`` and
      3   + ``]``). Square brackets are only valid for IPv6 and IPvFuture hosts according to
      4   + `RFC 3986 Section 3.2.2 <https://www.rfc-editor.org/rfc/rfc3986#section-
            3.2.2>`__.
```

Comments 0