

python / cpython Public

<> Code Issues 5k+ Pull requests 2.1k Actions Projects Security and q

# Commit a7084f6



3 people authored on Feb 2, 2025 Partially verified

```
[3.12] gh-105704: Disallow square brackets ([ and ]) in domain names for parsed
URLs (GH-129418) (GH-129527)

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

* gh-105704: Disallow square brackets ( and ) in domain names for parsed URLs

* Use Sphinx references

* Add mismatched bracket test cases, fix news format

* Add more test coverage for ports

-----

(cherry picked from commit d89a5f6)

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

3.12 (#129527) · v3.12.13 ... v3.12.9



1 parent 368ba55 commit a7084f6

**3 files changed** +58 -3

↑ Top ⚙️

🔍 Filter files...

- ✓ Lib
  - ✓ test
    - test\_urlparse.py
  - ✓ urllib

||  parse.pyv  Misc/NEWS.d/next/Security|  2025-01-28-14-08-03.gh-issue-105704.EnhHxu.rst

Q Search within code

v Lib/test/test\_urlparse.py ...

```
    ...    @@ -1273,16 +1273,51 @@ def test_invalid_bracketed_hosts(self):
1273 1273         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'Scheme://user@[0439:23af::2309::fae7:1234]/Path?Query')
1274 1274         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'Scheme://user@[0439:23af:2309::fae7:1234:2342:438e:192.0.2.146]/Path?Query')
1275 1275         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'Scheme://user@[v6a.ip]/Path')
1276 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://prefix.[v6a.ip]')
1277 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://[v6a.ip].suffix')
1278 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://prefix.[v6a.ip]/')
1279 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://[v6a.ip].suffix/')
1280 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://prefix.[v6a.ip]?')
1281 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://[v6a.ip].suffix?')
1282 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://prefix.[::1]')
1283 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://[::1].suffix')
1284 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://prefix.[::1]/')
1285 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://[::1].suffix/')
1286 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://prefix.[::1]?')
1287 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
        'scheme://[::1].suffix?')
```

```
1288 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1289 +                             'scheme://prefix.[::1]:a')
1290 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1291 +                             'scheme://[::1].suffix:a')
1292 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1293 +                             'scheme://prefix.[::1]:a1')
1294 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1295 +                             'scheme://[::1].suffix:a1')
1296 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1297 +                             'scheme://prefix.[::1]:1a')
1298 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1299 +                             'scheme://[::1].suffix:1a')
1300 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1301 +                             'scheme://prefix.[::1]:')
1302 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1303 +                             'scheme://[::1].suffix:/')
1304 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1305 +                             'scheme://prefix.[::1]:?')
1306 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1307 +                             'scheme://user@prefix.[v6a.ip]')
1308 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1309 +                             'scheme://user@[v6a.ip].suffix')
1310 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1311 +                             'scheme://[v6a.ip]')
1312 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1313 +                             'scheme://v6a.ip]')
1314 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1315 +                             'scheme://]v6a.ip[')
1316 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1317 +                             'scheme://]v6a.ip')
1318 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1319 +                             'scheme://v6a.ip[')
1320 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1321 +                             'scheme://prefix.[v6a.ip]')
1322 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1323 +                             'scheme://v6a.ip].suffix')
1324 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1325 +                             'scheme://prefix]v6a.ip[suffix')
1326 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1327 +                             'scheme://prefix]v6a.ip')
```

```

1308 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
                               'scheme://v6a.ip[suffix']
1276 1309
1277 1310         def test_splitting_bracketed_hosts(self):
1278 -         p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]/path?query')
1311 +         p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]:1234/path?query')
1279 1312         self.assertEqual(p1.hostname, 'v6a.ip')
1280 1313         self.assertEqual(p1.username, 'user')
1281 1314         self.assertEqual(p1.path, '/path')
1315 +         self.assertEqual(p1.port, 1234)
1282 1316         p2 =
                               urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7%test]/path?query')
1283 1317         self.assertEqual(p2.hostname, '0439:23af:2309::fae7%test')
1284 1318         self.assertEqual(p2.username, 'user')
1285 1319         self.assertEqual(p2.path, '/path')
1320 +         self.assertIs(p2.port, None)
1286 1321         p3 =
                               urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7:1234:192.0.2.146%t
est]/path?query')
1287 1322         self.assertEqual(p3.hostname,
                               '0439:23af:2309::fae7:1234:192.0.2.146%test')
1288 1323         self.assertEqual(p3.username, 'user')

```

Lib/urllib/parse.py

```

@@ -436,6 +436,23 @@ def _checknetloc(netloc):
436 436         raise ValueError("netloc '" + netloc + "' contains invalid " +
437 437             "characters under NFKC normalization")
438 438
439 + def _check_bracketed_netloc(netloc):
440 +     # Note that this function must mirror the splitting
441 +     # done in NetlocResultMixins._hostinfo().
442 +     hostname_and_port = netloc.rpartition('@')[2]
443 +     before_bracket, have_open_br, bracketed = hostname_and_port.partition('[')
444 +     if have_open_br:
445 +         # No data is allowed before a bracket.
446 +         if before_bracket:
447 +             raise ValueError("Invalid IPv6 URL")
448 +         hostname, _, port = bracketed.partition(']')
449 +         # No data is allowed after the bracket but before the port delimiter.

```

```

450 +         if port and not port.startswith(":"):
451 +             raise ValueError("Invalid IPv6 URL")
452 +         else:
453 +             hostname, _, port = hostname_and_port.partition(':')
454 +             _check_bracketed_host(hostname)
455 +
439 456 # Valid bracketed hosts are defined in
440 457 # https://www.rfc-editor.org/rfc/rfc3986#page-49 and
      https://url.spec.whatwg.org/
441 458 def _check_bracketed_host(hostname):
      @@ -496,8 +513,7 @@ def urlsplit(url, scheme='', allow_fragments=True):
      (']' in netloc and '[' not in netloc)):
496 513         raise ValueError("Invalid IPv6 URL")
497 514         if '[' in netloc and ']' in netloc:
499 -             bracketed_host = netloc.partition('[')[2].partition(']')[0]
500 -             _check_bracketed_host(bracketed_host)
516 +             _check_bracketed_netloc(netloc)
501 517         if allow_fragments and '#' in url:
502 518             url, fragment = url.split('#', 1)
503 519         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