

python / cpython Public

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

Commit 526617e



3 people authored on Feb 19, 2025 Partially verified



[3.11] gh-105704: Disallow square brackets ([and]) in domain names for parsed URLs (GH-129418) (#129528)

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

3.11 (#98846, #129528) · v3.11.15 ... v3.11.12

1 parent 7cff053 commit 526617e

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

```

@@ -1224,16 +1224,51 @@ def test_invalid_bracketed_hosts(self):
1224 1224         self.assertRaises(ValueError, urllib.parse.urlsplit,
                'Scheme://user@[0439:23af::2309::fae7:1234]/Path?Query')
```

```
1225 1225         self.assertRaises(ValueError, urllib.parse.urlsplit,
'Scheme://user@[0439:23af:2309::fae7:1234:2342:438e:192.0.2.146]/Path?Query')
1226 1226         self.assertRaises(ValueError, urllib.parse.urlsplit,
'Scheme://user@[v6a.ip]/Path')
1227 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[v6a.ip]')
1228 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[v6a.ip].suffix')
1229 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[v6a.ip]/')
1230 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[v6a.ip].suffix/')
1231 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[v6a.ip]?')
1232 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[v6a.ip].suffix?')
1233 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]')
1234 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix')
1235 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]/')
1236 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix/')
1237 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]?')
1238 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix?')
1239 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]:a')
1240 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix:a')
1241 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]:a1')
1242 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix:a1')
1243 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://prefix.[::1]:1a')
1244 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
'scheme://[::1].suffix:1a')
```

```
1245 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1246 +                             'scheme://prefix.[::1]:')
1247 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1248 +                             'scheme://[::1].suffix/')
1249 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1250 +                             'scheme://prefix.[::1]:?')
1251 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1252 +                             'scheme://user@prefix.[v6a.ip]')
1253 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1254 +                             'scheme://user@[v6a.ip].suffix')
1255 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1256 +                             'scheme://[v6a.ip]')
1257 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1258 +                             'scheme://v6a.ip]')
1259 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1260 +                             'scheme://]v6a.ip[')
1261 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1262 +                             'scheme://]v6a.ip')
1263 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1264 +                             'scheme://v6a.ip[')
1265 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1266 +                             'scheme://prefix.[v6a.ip]')
1267 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1268 +                             'scheme://v6a.ip].suffix')
1269 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1270 +                             'scheme://prefix]v6a.ip[suffix')
1271 +         self.assertRaises(ValueError, urllib.parse.urlsplit,
1272 +                             'scheme://prefix]v6a.ip')
```

```
1227 1260
```

```
1228 1261         def test_splitting_bracketed_hosts(self):
```

```
1229 -         p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]/path?query')
```

```
1262 +         p1 = urllib.parse.urlsplit('scheme://user@[v6a.ip]:1234/path?query')
```

```
1230 1263         self.assertEqual(p1.hostname, 'v6a.ip')
```

```
1231 1264         self.assertEqual(p1.username, 'user')
```

```
1232 1265         self.assertEqual(p1.path, '/path')
```

```
1266 +         self.assertEqual(p1.port, 1234)
```

```
1233 1267
```

```
        p2 =
```

```
        urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7%test]/path?query')
```

```

1234 1268         self.assertEqual(p2.hostname, '0439:23af:2309::fae7%test')
1235 1269         self.assertEqual(p2.username, 'user')
1236 1270         self.assertEqual(p2.path, '/path')
1271 +         self.assertIs(p2.port, None)
1237 1272         p3 =
        urllib.parse.urlsplit('scheme://user@[0439:23af:2309::fae7:1234:192.0.2.146%t
est]/path?query')
1238 1273         self.assertEqual(p3.hostname,
        '0439:23af:2309::fae7:1234:192.0.2.146%test')
1239 1274         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):

```

```
496 513         (']' in netloc and '[' not in netloc)):
497 514         raise ValueError("Invalid IPv6 URL")
498 515         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