python / **cpython** Public

Code Issues 5k+ Pull requests 2.2k Actions Projects Security and q

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

Commit d4df3c5

4 people authored on Jun 2, 2025 · ✕ 12 / 15 · Partially verified

```
[3.9] gh-80222: Fix email address header folding with long quoted-string (GH-
122753) (GH-129111) (GH-132371)

Email generators using email.policy.default could incorrectly omit the
quote ('"') characters from a quoted-string during header refolding,
leading to invalid address headers and enabling header spoofing. This
change restores the quote characters on a bare-quoted-string as the
header is refolded, and escapes backslash and quote chars in the string.

(cherry picked from commit 5aaf416)
(cherry picked from commit a4ef689)

Co-authored-by: R. David Murray <rdmurray@bitdance.com>
Co-authored-by: Mike Edmunds <medmunds@gmail.com>
Co-authored-by: Łukasz Langa <lukasz@langa.pl>
```

· v3.9.25 ••• 3.9

1 parent 558e27a commit d4df3c5

3 files changed +51 -1

Top

Filter files…

- Lib
  - email
    - _header_value_parser.py
  - test/test_email

test__header_value_parser.py

Misc/NEWS.d/next/Security

2024-08-06-11-43-08.gh-issue-80222.wfR4BU.rst

Search within code

Lib/email/_header_value_parser.py

```
@@ -95,8 +95,16 @@
95  95  NLSET = {'\n', '\r'}
96  96  SPECIALSNL = SPECIALS | NLSET
97  97
    98  +
    99  + def make_quoted_pairs(value):
   100  +     """Escape dquote and backslash for use within a quoted-string."""
   101  +     return str(value).replace('\\', '\\\\').replace('"', '\\"')
   102  +
   103  +
98  104 def quote_string(value):
99      -     return '"'+str(value).replace('\\', '\\\\').replace('"', r'\"')+'"'
    105 +     escaped = make_quoted_pairs(value)
    106 +     return f'"{escaped}"'
    107 +
100 108
101 109 # Match a RFC 2047 word, looks like =?utf-8?q?someword?=
102 110 rfc2047_matcher = re.compile(r'''
@@ -2848,6 +2856,15 @@ def _refold_parse_tree(parse_tree, *, policy):
2848 2856             if not hasattr(part, 'encode'):
2849 2857                 # It's not a terminal, try folding the subparts.
2850 2858                 newparts = list(part)
     2859 +               if part.token_type == 'bare-quoted-string':
     2860 +                   # To fold a quoted string we need to create a list of
       terminal
     2861 +                   # tokens that will render the leading and trailing quotes
     2862 +                   # and use quoted pairs in the value as appropriate.
     2863 +                   newparts = (
     2864 +                       [ValueTerminal('"', 'ptext')] +
     2865 +                       [ValueTerminal(make_quoted_pairs(p), 'ptext')
```

```
        2866 +                     for p in newparts] +
        2867 +                    [ValueTerminal('"', 'ptext')])
2851    2868                 if not part.as_ew_allowed:
2852    2869                     wrap_as_ew_blocked += 1
2853    2870                     newparts.append(end_ew_not_allowed)
```

…est/test_email/test__header_value_parser.py

```
@@ -2946,6 +2946,33 @@ def
test_address_list_with_unicode_names_in_quotes(self):
2946    2946                 '=?utf-8?q?H=C3=BCbsch?= Kaktus <beautiful@example.com>,\n'
2947    2947                     ' =?utf-8?q?bei=C3=9Ft_bei=C3=9Ft?= <biter@example.com>\n')
2948    2948
        2949 +     def test_address_list_with_specials_in_long_quoted_string(self):
        2950 +         # Regression for gh-80222.
        2951 +         policy = self.policy.clone(max_line_length=40)
        2952 +         cases = [
        2953 +             # (to, folded)
        2954 +             ('"Exfiltrator <spy@example.org> (unclosed comment?"
<to@example.com>',
        2955 +              '"Exfiltrator <spy@example.org> (unclosed\n'
        2956 +              ' comment?" <to@example.com>\n'),
        2957 +             ('"Escaped \\" chars \\\\ in quoted-string stay escaped"
<to@example.com>',
        2958 +              '"Escaped \\" chars \\\\ in quoted-string\n'
        2959 +              ' stay escaped" <to@example.com>\n'),
        2960 +             ('This long display name does not need quotes <to@example.com>',
        2961 +              'This long display name does not need\n'
        2962 +              ' quotes <to@example.com>\n'),
        2963 +             ('"Quotes are not required but are retained here"
<to@example.com>',
        2964 +              '"Quotes are not required but are\n'
        2965 +              ' retained here" <to@example.com>\n'),
        2966 +             ('"A quoted-string, it can be a valid local-part"@example.com',
        2967 +              '"A quoted-string, it can be a valid\n'
        2968 +              ' local-part"@example.com\n'),
        2969 +             ('"local-part-with-specials@but-no-fws.cannot-fold"@example.com',
        2970 +              '"local-part-with-specials@but-no-fws.cannot-
fold"@example.com\n'),
        2971 +         ]
```

```
        2972 +         for (to, folded) in cases:
        2973 +             with self.subTest(to=to):
        2974 +                 self._test(parser.get_address_list(to)[0], folded,
                   policy=policy)
        2975 +
2949    2976         def test_address_list_with_specials_in_encoded_word(self):
2950    2977             # An encoded-word parsed from a structured header must remain
2951    2978             # encoded when it contains specials. Regression for gh-121284.
```

...24-08-06-11-43-08.gh-issue-80222.wfR4BU.rst

```
        @@ -0,0 +1,6 @@
1       + Fix bug in the folding of quoted strings when flattening an email message using
2       + a modern email policy. Previously when a quoted string was folded so that
3       + it spanned more than one line, the surrounding quotes and internal escapes
4       + would be omitted. This could theoretically be used to spoof header lines
5       + using a carefully constructed quoted string if the resulting rendered email
6       + was transmitted or re-parsed.
```

Comments 0