Commit

✓ [3.8] gh-123270: Replaced SanitizedNames with a more surgical fix. (G...

Browse files

...H-123354) (#123433)

```
Applies changes from zipp 3.20.1 and jaraco/zippGH-124
(cherry picked from commit 2231286)
(cherry picked from commit 17b77bb)
(cherry picked from commit 66d3383)

Co-authored-by: Jason R. Coombs <jaraco@jaraco.com>
```

3.8 (#123433)

jaraco committed yesterday Verified 1 parent a77ab24 commit 7bc367e

Showing **3 changed files** with **87 additions** and **2 deletions**.

Whitesp... Ignore whitespa... Sp... Unifi...

Filter changed files

- Lib
  - test
    - test_zipfile.py
    - zipfile.py
- Misc/NEWS.d/next/Library
  - 2024-08-26-13-45-2...

77 Lib/test/test_zipfile.py

```
3007 3007          data = ['/'.join(string.ascii_lowercase + str(n)) for n in range(10000)]
3008 3008
     3009          zipfile.CompleteDirs._implied_dirs(data)
3009 3010
     3011  +    def test_malformed_paths(self):
     3012  +        """
     3013  +        Path should handle malformed paths gracefully.
     3014  +
     3015  +        Paths with leading slashes are not visible.
     3016  +
     3017  +        Paths with dots are treated like regular files.
     3018  +        """
     3019  +        data = io.BytesIO()
     3020  +        zf = zipfile.ZipFile(data, "w")
     3021  +        zf.writestr("/one-slash.txt", b"content")
     3022  +        zf.writestr("//two-slash.txt", b"content")
     3023  +        zf.writestr("../parent.txt", b"content")
     3024  +        zf.filename = ''
     3025  +        root = zipfile.Path(zf)
```

```
3025 +         assert list(map(str, root.iterdir())) == ['../']
3026 +         assert root.joinpath('..').joinpath('parent.txt').read_bytes() == b'content'
3027 +
3028 +     def test_unsupported_names(self):
3029 +         """
3030 +         Path segments with special characters are readable.
3031 +
3032 +         On some platforms or file systems, characters like
3033 +         ``:`` and ``?`` are not allowed, but they are valid
3034 +         in the zip file.
3035 +         """
3036 +         data = io.BytesIO()
3037 +         zf = zipfile.ZipFile(data, "w")
3038 +         zf.writestr("path?", b"content")
3039 +         zf.writestr("V: NMS.flac", b"fLaC...")
3040 +         zf.filename = ''
3041 +         root = zipfile.Path(zf)
3042 +         contents = root.iterdir()
3043 +         assert next(contents).name == 'path?'
3044 +         assert next(contents).name == 'V: NMS.flac'
3045 +         assert root.joinpath('V: NMS.flac').read_bytes() == b"fLaC..."
3046 +
3047 +     def test_backslash_not_separator(self):
3048 +         """
3049 +         In a zip file, backslashes are not separators.
3050 +         """
3051 +         data = io.BytesIO()
3052 +         zf = zipfile.ZipFile(data, "w")
3053 +         zf.writestr(DirtyZipInfo.for_name("foo\\bar", zf), b"content")
3054 +         zf.filename = ''
3055 +         root = zipfile.Path(zf)
3056 +         (first,) = root.iterdir()
3057 +         assert not first.is_dir()
3058 +         assert first.name == 'foo\\bar'
3059 +
3060 +
3061 + class DirtyZipInfo(zipfile.ZipInfo):
3062 +     """
3063 +     Bypass name sanitization.
3064 +     """
3065 +
3066 +     def __init__(self, filename, *args, **kwargs):
3067 +         super().__init__(filename, *args, **kwargs)
```

```
3068 +         self.filename = filename
3069 +
3070 +     @classmethod
3071 +     def for_name(cls, name, archive):
3072 +         """
3073 +         Construct the same way that
     ZipFile.writestr does.
3074 +
3075 +         TODO: extract this functionality and
     re-use
3076 +         """
3077 +         self = cls(filename=name,
     date_time=time.localtime(time.time())[:6])
3078 +         self.compress_type =
     archive.compression
3079 +         self.compress_level =
     archive.compresslevel
3080 +         if self.filename.endswith('/'):  #
     pragma: no cover
3081 +             self.external_attr = 0o40775 << 16
     # drwxrwxr-x
3082 +             self.external_attr |= 0x10  # MS-
     DOS directory flag
3083 +         else:
3084 +             self.external_attr = 0o600 << 16  #
     ?rw-------
3085 +         return self
3086 +
3010 3087
3011 3088   if __name__ == "__main__":
3012 3089       unittest.main()
```

9 ▪▪▪▪▫ Lib/zipfile.py

```
2161 2161   def _ancestry(path):
2162 2162       """
2163 2163       Given a path with elements separated by
2164      -     posixpath.sep, generate all elements of
     that path
     2164 +     posixpath.sep, generate all elements of
     that path.
2165 2165
2166 2166       >>> list(_ancestry('b/d'))
2167 2167       ['b/d', 'b']
2173 2173       ['b']
2174 2174       >>> list(_ancestry(''))
2175 2175       []
     2176 +
     2177 +     Multiple separators are treated like a
     single.
     2178 +
     2179 +     >>> list(_ancestry('//b//d///f//'))
     2180 +     ['//b//d///f', '//b//d', '//b']
2176 2181       """
2177 2182       path = path.rstrip(posixpath.sep)
2178      -     while path and path != posixpath.sep:
     2183 +     while path.rstrip(posixpath.sep):
```

```
2179  2184            yield path
2180  2185            path, tail = posixpath.split(path)
2181  2186
```

3 ▪▪▪▫▫

Misc/NEWS.d/next/Library/2024-08-26-13-45-20.gh-issue-1232…

```
...   ...   @@ -0,0 +1,3 @@
      1   + Applied a more surgical fix for malformed payloads
            in :class:`zipfile.Path`
      2   + causing infinite loops (gh-122905) without
            breaking contents using
      3   + legitimate characters.
```

0 comments on commit `7bc367e`

Please sign in to comment.