Commit

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

Browse files

...H-123354) (#123425)

Applies changes from zipp 3.20.1 and jaraco/zippGH-124
(cherry picked from commit 2231286)

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

* Restore the slash-prefixed paths in the malformed_paths test.

3.11 (#98846, #123425)

jaraco committed yesterday Verified

1 parent d4ac921 commit fc0b825

Showing **3 changed files** with **77 additions** and **67 deletions**.

Whitesp... Ignore whitesp... S... Uni...

Filter changed files

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

72 Lib/test/test_zipfile.py

```
3653 3653
3654 3654       def test_malformed_paths(self):
3655 3655           """
3656      -         Path should handle malformed paths.
     3656 +         Path should handle malformed paths gracefully.
     3657 +
     3658 +         Paths with leading slashes are not visible.
     3659 +
     3660 +         Paths with dots are treated like regular files.
3657 3661           """
3658 3662           data = io.BytesIO()
3659 3663           zf = zipfile.ZipFile(data, "w")
3662 3666           zf.writestr("../parent.txt", b"content")
3663 3667           zf.filename = ''
3664 3668           root = zipfile.Path(zf)
3665      -         assert list(map(str, root.iterdir())) == [
3666      -             'one-slash.txt',
3667      -             'two-slash.txt',
3668      -             'parent.txt',
3669      -         ]
```

```
3669 +         assert list(map(str, root.iterdir())) == ['../']
3670 +         assert root.joinpath('..').joinpath('parent.txt').read_bytes() == b'content'
3671 +
3672 +     def test_unsupported_names(self):
3673 +         """
3674 +         Path segments with special characters are readable.
3675 +
3676 +         On some platforms or file systems, characters like
3677 +         ``:`` and ``?`` are not allowed, but they are valid
3678 +         in the zip file.
3679 +         """
3680 +         data = io.BytesIO()
3681 +         zf = zipfile.ZipFile(data, "w")
3682 +         zf.writestr("path?", b"content")
3683 +         zf.writestr("V: NMS.flac", b"fLaC...")
3684 +         zf.filename = ''
3685 +         root = zipfile.Path(zf)
3686 +         contents = root.iterdir()
3687 +         assert next(contents).name == 'path?'
3688 +         assert next(contents).name == 'V: NMS.flac'
3689 +         assert root.joinpath('V: NMS.flac').read_bytes() == b"fLaC..."
3690 +
3691 +     def test_backslash_not_separator(self):
3692 +         """
3693 +         In a zip file, backslashes are not separators.
3694 +         """
3695 +         data = io.BytesIO()
3696 +         zf = zipfile.ZipFile(data, "w")
3697 +         zf.writestr(DirtyZipInfo.for_name("foo\\bar", zf), b"content")
3698 +         zf.filename = ''
3699 +         root = zipfile.Path(zf)
3700 +         (first,) = root.iterdir()
3701 +         assert not first.is_dir()
3702 +         assert first.name == 'foo\\bar'
3703 +
3704 +
3705 + class DirtyZipInfo(zipfile.ZipInfo):
3706 +     """
3707 +     Bypass name sanitization.
3708 +     """
3709 +
3710 +     def __init__(self, filename, *args, **kwargs):
3711 +         super().__init__(filename, *args, **kwargs)
```

```
3712 +         self.filename = filename
3713 +
3714 +     @classmethod
3715 +     def for_name(cls, name, archive):
3716 +         """
3717 +         Construct the same way that
     ZipFile.writestr does.
3718 +
3719 +         TODO: extract this functionality and
     re-use
3720 +         """
3721 +         self = cls(filename=name,
     date_time=time.localtime(time.time())[:6])
3722 +         self.compress_type =
     archive.compression
3723 +         self.compress_level =
     archive.compresslevel
3724 +         if self.filename.endswith('/'):  #
     pragma: no cover
3725 +             self.external_attr = 0o40775 << 16
     # drwxrwxr-x
3726 +             self.external_attr |= 0x10  # MS-
     DOS directory flag
3727 +         else:
3728 +             self.external_attr = 0o600 << 16  #
     ?rw-------
3729 +         return self
3670 3730
3671 3731
3672 3732  class EncodedMetadataTests(unittest.TestCase):
```

69 ▪▪▪▪▫ Lib/zipfile.py

```
2213 2213  def _ancestry(path):
2214 2214      """
2215 2215      Given a path with elements separated by
2216      -     posixpath.sep, generate all elements of
     that path
     2216 +     posixpath.sep, generate all elements of
     that path.
2217 2217
2218 2218      >>> list(_ancestry('b/d'))
2219 2219      ['b/d', 'b']
2225 2225      ['b']
2226 2226      >>> list(_ancestry(''))
2227 2227      []
     2228 +
     2229 +     Multiple separators are treated like a
     single.
     2230 +
     2231 +     >>> list(_ancestry('//b//d///f//'))
     2232 +     ['//b//d///f', '//b//d', '//b']
2228 2233      """
2229 2234      path = path.rstrip(posixpath.sep)
2230      -     while path and path != posixpath.sep:
     2235 +     while path.rstrip(posixpath.sep):
2231 2236          yield path
```

```
2232  2237          path, tail = posixpath.split(path)
2233  2238
2244  2249      return
      itertools.filterfalse(set(subtrahend).__contain
      s__, minuend)
2245  2250
2246  2251
2247        - class SanitizedNames:
2248        -     """
2249        -     ZipFile mix-in to ensure names are
      sanitized.
2250        -     """
2251        -
2252        -     def namelist(self):
2253        -         return list(map(self._sanitize,
      super().namelist()))
2254        -
2255        -     @staticmethod
2256        -     def _sanitize(name):
2257        -         r"""
2258        -         Ensure a relative path with posix
      separators and no dot names.
2259        -         Modeled after
2260        -
      https://github.com/python/cpython/blob/bcc1be39
      cb1d04ad9fc0bd1b9193d3972835a57c/Lib/zipfile/__
      init__.py#L1799-L1813
2261        -         but provides consistent cross-platform
      behavior.
2262        -         >>> san = SanitizedNames._sanitize
2263        -         >>> san('/foo/bar')
2264        -         'foo/bar'
2265        -         >>> san('//foo.txt')
2266        -         'foo.txt'
2267        -         >>> san('foo/.././bar.txt')
2268        -         'foo/bar.txt'
2269        -         >>> san('foo../.bar.txt')
2270        -         'foo../.bar.txt'
2271        -         >>> san('\\foo\\bar.txt')
2272        -         'foo/bar.txt'
2273        -         >>> san('D:\\foo.txt')
2274        -         'D/foo.txt'
2275        -         >>> san('\\\\server\\share\\file.txt')
2276        -         'server/share/file.txt'
2277        -         >>> san('\\\\?\\GLOBALROOT\\Volume3')
2278        -         '?/GLOBALROOT/Volume3'
2279        -         >>> san('\\\\.\\PhysicalDrive1\\root')
2280        -         'PhysicalDrive1/root'
2281        -         Retain any trailing slash.
2282        -         >>> san('abc/')
2283        -         'abc/'
2284        -         Raises a ValueError if the result is
      empty.
2285        -         >>> san('../..')
2286        -         Traceback (most recent call last):
2287        -         ...
```

```
2288          -         ValueError: Empty filename
2289          -         """
2290          -
2291          -         def allowed(part):
2292          -             return part and part not in {'..',
                '.'}
2293          -
2294          -         # Remove the drive letter.
2295          -         # Don't use ntpath.splitdrive, because
                that also strips UNC paths
2296          -         bare = re.sub('^([A-Z]):', r'\1', name,
                flags=re.IGNORECASE)
2297          -         clean = bare.replace('\\', '/')
2298          -         parts = clean.split('/')
2299          -         joined = '/'.join(filter(allowed,
                parts))
2300          -         if not joined:
2301          -             raise ValueError("Empty filename")
2302          -         return joined + '/' *
                name.endswith('/')
2303          -
2304          -
2305          - class CompleteDirs(SanitizedNames, ZipFile):
      2252    + class CompleteDirs(ZipFile):
2306  2253            """
2307  2254            A ZipFile subclass that ensures that
                implied directories
2308  2255            are always included in the namelist.
```

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 `fc0b825`

Please sign in to comment.