

Commit

✖ gh-123270: Replaced SanitizedNames with a more surgical fix. (#123354)

Browse files

Applies changes from zipp 3.20.1 and [jaraco/zipp#124](#)

main (#123354)

 jaraco committed last week Verified

1 parent 7e38e67

commit 2231286

Showing 3 changed files with 87 additions and 71 deletions.

Whitesp...

Ignore whitesp...

S...

Uni...

Filter changed files

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

73 Lib/test/test_zipfile/_path/test_path.py

```
5      5      import pickle
6      6      import stat
7      7      import sys
8      8      + import time
9      9      import unittest
10     10      import zipfile
10     11      import zipfile._path
592   593
593   594      def test_malformed_paths(self):
594   595          """
595   596          - Path should handle malformed paths.
596   597          + Path should handle malformed paths gracefully.
597   598          +
598   599          + Paths with leading slashes are not visible.
599   600          +
600   601          + Paths with dots are treated like regular files.
601   602          """
602   603          data = io.BytesIO()
603   604          zf = zipfile.ZipFile(data, "w")
604   605          zf.writestr("../parent.txt",
605   606                      b"content")
606   607          zf.filename = ''
607   608          root = zipfile.Path(zf)
608   609          assert list(map(str, root.iterdir()))
609   610          == [
610   611              'one-slash.txt',
611   612              'two-slash.txt',
612   613              'parent.txt',
613   614          ]
```

```

609 +         assert list(map(str, root.iterdir()))
        == ['../']
610 +         assert
        root.joinpath('..').joinpath('parent.txt').read
        _bytes() == b'content'
611 +
612 +         def test_unsupported_names(self):
613 +             """
614 +             Path segments with special characters
        are readable.
615 +
616 +             On some platforms or file systems,
        characters like
617 +             ``:`` and ``?`` are not allowed, but
        they are valid
618 +             in the zip file.
619 +             """
620 +             data = io.BytesIO()
621 +             zf = zipfile.ZipFile(data, "w")
622 +             zf.writestr("path?", b"content")
623 +             zf.writestr("V: NMS.flac", b"fLaC...")
624 +             zf.filename = ''
625 +             root = zipfile.Path(zf)
626 +             contents = root.iterdir()
627 +             assert next(contents).name == 'path?'
628 +             assert next(contents).name == 'V:
        NMS.flac'
629 +             assert root.joinpath('V:
        NMS.flac').read_bytes() == b"fLaC..."
630 +
631 +         def test_backslash_not_separator(self):
632 +             """
633 +             In a zip file, backslashes are not
        separators.
634 +             """
635 +             data = io.BytesIO()
636 +             zf = zipfile.ZipFile(data, "w")
637 +
        zf.writestr(DirtyZipInfo.for_name("foo\\bar",
        zf), b"content")
638 +             zf.filename = ''
639 +             root = zipfile.Path(zf)
640 +             (first,) = root.iterdir()
641 +             assert not first.is_dir()
642 +             assert first.name == 'foo\\bar'
609 643
610 644         @pass_alpharep
611 645         def test_interface(self, alpharep):
612 646             from importlib.resources.abc import
        Traversable
613 647
614 648             zf = zipfile.Path(alpharep)
615 649             assert isinstance(zf, Traversable)
650 +
651 +
652 + class DirtyZipInfo(zipfile.ZipInfo):

```

```

653 + """
654 +     Bypass name sanitization.
655 +     """
656 +
657 +     def __init__(self, filename, *args,
658 +                  **kwargs):
659 +         super().__init__(filename, *args,
660 +                          **kwargs)
661 +         self.filename = filename
662 +
663 +         @classmethod
664 +         def for_name(cls, name, archive):
665 +             """
666 +             Construct the same way that
667 +             ZipFile.writestr does.
668 +
669 +             TODO: extract this functionality and
670 +             re-use
671 +             """
672 +             self = cls(filename=name,
673 +                        date_time=time.localtime(time.time())[:6])
674 +             self.compress_type =
675 +                 archive.compression
676 +             self.compress_level =
677 +                 archive.compresslevel
678 +             if self.filename.endswith('/'): #
679 +                 pragma: no cover
680 +                 self.external_attr = 0o40775 << 16
681 +                 # drwxrwxr-x
682 +                 self.external_attr |= 0x10 # MS-
683 +                 DOS directory flag
684 +             else:
685 +                 self.external_attr = 0o600 << 16 #
686 +                 ?rw-----
687 +             return self

```



82   Lib/zipfile/_path/__init__.py 

```

...     @@ -1,3 +1,12 @@
...     1 + """
...     2 + A Path-like interface for zipfiles.
...     3 +
...     4 + This codebase is shared between zipfile.Path in
...     5 + the stdlib
...     6 + and zipp in PyPI. See
...     7 + https://github.com/python/importlib_metadata/wiki/Development-Methodology
...     8 + for more detail.
...     9 + """
...    10 +
...    11 + import io
...    12 + import posixpath
...    13 + import zipfile
...    14 +
...    15 + def _ancestry(path):
...    16 +     """
...    17 +     Given a path with elements separated by

```

```

39         posixpath.sep, generate all elements of
40         that path
41     48 +     posixpath.sep, generate all elements of
42         that path.
43     49
44     50 >>> list(_ancestry('b/d'))
45     51 ['b/d', 'b']
46     52 ['b']
47     53 >>> list(_ancestry(''))
48     54 []
49     55
50     56 +
51     57 +     Multiple separators are treated like a
52         single.
53     58
54     59 +
55     60 >>> list(_ancestry('//b//d//f//'))
56     61 ['//b//d//f', '//b//d', '//b']
57     62 ""
58     63
59     64 path = path.rstrip(posixpath.sep)
60     65
61     66 while path and path != posixpath.sep:
62     67 while path.rstrip(posixpath.sep):
63     68     yield path
64     69     path, tail = posixpath.split(path)
65     70
66     71
67     72 super().__init__(*args, **kwargs)
68     73
69     74
70     75
71     76
72     77
73     78
74     79
75     80
76     81
77     82
78     83
79     84
80     85
81     86
82     87
83     88 - class SanitizedNames:
84     89 -     """
85     90 -     ZipFile mix-in to ensure names are
86         sanitized.
87     91 -     """
88     92 -
89     93 -     def namelist(self):
90     94 -     return list(map(self._sanitize,
91         super().namelist()))
92     95 -
93     96 -     @staticmethod
94     97 -     def _sanitize(name):
95     98 -     r"""
96     99 -     Ensure a relative path with posix
97     100 separators and no dot names.
98     101
99     102
100     103
101     104
102     105
103     106
104     107
105     108
106     109
107     110
108     111
109     112
110     113
111     114
112     115
113     116
114     117
115     118
116     119
117     120
118     121
119     122
120     123
121     124
122     125
123     126
124     127
125     128
126     129
127     130
128     131
129     132
130     133
131     134
132     135
133     136
134     137
135     138
136     139
137     140
138     141
139     142
140     143
141     144
142     145
143     146
144     147
145     148
146     149
147     150
148     151
149     152
150     153
151     154
152     155
153     156
154     157
155     158
156     159
157     160
158     161
159     162
160     163
161     164
162     165
163     166
164     167
165     168
166     169
167     170
168     171
169     172
170     173
171     174
172     175
173     176
174     177
175     178
176     179
177     180
178     181
179     182
180     183
181     184
182     185
183     186
184     187
185     188
186     189
187     190
188     191
189     192
190     193
191     194
192     195
193     196
194     197
195     198
196     199
197     200
198     201
199     202
200     203
201     204
202     205
203     206
204     207
205     208
206     209
207     210
208     211
209     212
210     213
211     214
212     215
213     216
214     217
215     218
216     219
217     220
218     221
219     222
220     223
221     224
222     225
223     226
224     227
225     228
226     229
227     230
228     231
229     232
230     233
231     234
232     235
233     236
234     237
235     238
236     239
237     240
238     241
239     242
240     243
241     244
242     245
243     246
244     247
245     248
246     249
247     250
248     251
249     252
250     253
251     254
252     255
253     256
254     257
255     258
256     259
257     260
258     261
259     262
260     263
261     264
262     265
263     266
264     267
265     268
266     269
267     270
268     271
269     272
270     273
271     274
272     275
273     276
274     277
275     278
276     279
277     280
278     281
279     282
280     283
281     284
282     285
283     286
284     287
285     288
286     289
287     290
288     291
289     292
290     293
291     294
292     295
293     296
294     297
295     298
296     299
297     300
298     301
299     302
300     303
301     304
302     305
303     306
304     307
305     308
306     309
307     310
308     311
309     312
310     313
311     314
312     315
313     316
314     317
315     318
316     319
317     320
318     321
319     322
320     323
321     324
322     325
323     326
324     327
325     328
326     329
327     330
328     331
329     332
330     333
331     334
332     335
333     336
334     337
335     338
336     339
337     340
338     341
339     342
340     343
341     344
342     345
343     346
344     347
345     348
346     349
347     350
348     351
349     352
350     353
351     354
352     355
353     356
354     357
355     358
356     359
357     360
358     361
359     362
360     363
361     364
362     365
363     366
364     367
365     368
366     369
367     370
368     371
369     372
370     373
371     374
372     375
373     376
374     377
375     378
376     379
377     380
378     381
379     382
380     383
381     384
382     385
383     386
384     387
385     388
386     389
387     390
388     391
389     392
390     393
391     394
392     395
393     396
394     397
395     398
396     399
397     400
398     401
399     402
400     403
401     404
402     405
403     406
404     407
405     408
406     409
407     410
408     411
409     412
410     413
411     414
412     415
413     416
414     417
415     418
416     419
417     420
418     421
419     422
420     423
421     424
422     425
423     426
424     427
425     428
426     429
427     430
428     431
429     432
430     433
431     434
432     435
433     436
434     437
435     438
436     439
437     440
438     441
439     442
440     443
441     444
442     445
443     446
444     447
445     448
446     449
447     450
448     451
449     452
450     453
451     454
452     455
453     456
454     457
455     458
456     459
457     460
458     461
459     462
460     463
461     464
462     465
463     466
464     467
465     468
466     469
467     470
468     471
469     472
470     473
471     474
472     475
473     476
474     477
475     478
476     479
477     480
478     481
479     482
480     483
481     484
482     485
483     486
484     487
485     488
486     489
487     490
488     491
489     492
490     493
491     494
492     495
493     496
494     497
495     498
496     499
497     500
498     501
499     502
500     503
501     504
502     505
503     506
504     507
505     508
506     509
507     510
508     511
509     512
510     513
511     514
512     515
513     516
514     517
515     518
516     519
517     520
518     521
519     522
520     523
521     524
522     525
523     526
524     527
525     528
526     529
527     530
528     531
529     532
530     533
531     534
532     535
533     536
534     537
535     538
536     539
537     540
538     541
539     542
540     543
541     544
542     545
543     546
544     547
545     548
546     549
547     550
548     551
549     552
550     553
551     554
552     555
553     556
554     557
555     558
556     559
557     560
558     561
559     562
560     563
561     564
562     565
563     566
564     567
565     568
566     569
567     570
568     571
569     572
570     573
571     574
572     575
573     576
574     577
575     578
576     579
577     580
578     581
579     582
580     583
581     584
582     585
583     586
584     587
585     588
586     589
587     590
588     591
589     592
590     593
591     594
592     595
593     596
594     597
595     598
596     599
597     600
598     601
599     602
600     603
601     604
602     605
603     606
604     607
605     608
606     609
607     610
608     611
609     612
610     613
611     614
612     615
613     616
614     617
615     618
616     619
617     620
618     621
619     622
620     623
621     624
622     625
623     626
624     627
625     628
626     629
627     630
628     631
629     632
630     633
631     634
632     635
633     636
634     637
635     638
636     639
637     640
638     641
639     642
640     643
641     644
642     645
643     646
644     647
645     648
646     649
647     650
648     651
649     652
650     653
651     654
652     655
653     656
654     657
655     658
656     659
657     660
658     661
659     662
660     663
661     664
662     665
663     666
664     667
665     668
666     669
667     670
668     671
669     672
670     673
671     674
672     675
673     676
674     677
675     678
676     679
677     680
678     681
679     682
680     683
681     684
682     685
683     686
684     687
685     688
686     689
687     690
688     691
689     692
690     693
691     694
692     695
693     696
694     697
695     698
696     699
697     700
698     701
699     702
700     703
701     704
702     705
703     706
704     707
705     708
706     709
707     710
708     711
709     712
710     713
711     714
712     715
713     716
714     717
715     718
716     719
717     720
718     721
719     722
720     723
721     724
722     725
723     726
724     727
725     728
726     729
727     730
728     731
729     732
730     733
731     734
732     735
733     736
734     737
735     738
736     739
737     740
738     741
739     742
740     743
741     744
742     745
743     746
744     747
745     748
746     749
747     750
748     751
749     752
750     753
751     754
752     755
753     756
754     757
755     758
756     759
757     760
758     761
759     762
760     763
761     764
762     765
763     766
764     767
765     768
766     769
767     770
768     771
769     772
770     773
771     774
772     775
773     776
774     777
775     778
776     779
777     780
778     781
779     782
780     783
781     784
782     785
783     786
784     787
785     788
786     789
787     790
788     791
789     792
790     793
791     794
792     795
793     796
794     797
795     798
796     799
797     800
798     801
799     802
800     803
801     804
802     805
803     806
804     807
805     808
806     809
807     810
808     811
809     812
810     813
811     814
812     815
813     816
814     817
815     818
816     819
817     820
818     821
819     822
820     823
821     824
822     825
823     826
824     827
825     828
826     829
827     830
828     831
829     832
830     833
831     834
832     835
833     836
834     837
835     838
836     839
837     840
838     841
839     842
840     843
841     844
842     845
843     846
844     847
845     848
846     849
847     850
848     851
849     852
850     853
851     854
852     855
853     856
854     857
855     858
856     859
857     860
858     861
859     862
860     863
861     864
862     865
863     866
864     867
865     868
866     869
867     870
868     871
869     872
870     873
871     874
872     875
873     876
874     877
875     878
876     879
877     880
878     881
879     882
880     883
881     884
882     885
883     886
884     887
885     888
886     889
887     890
888     891
889     892
890     893
891     894
892     895
893     896
894     897
895     898
896     899
897     900
898     901
899     902
900     903
901     904
902     905
903     906
904     907
905     908
906     909
907     910
908     911
909     912
910     913
911     914
912     915
913     916
914     917
915     918
916     919
917     920
918     921
919     922
920     923
921     924
922     925
923     926
924     927
925     928
926     929
927     930
928     931
929     932
930     933
931     934
932     935
933     936
934     937
935     938
936     939
937     940
938     941
939     942
940     943
941     944
942     945
943     946
944     947
945     948
946     949
947     950
948     951
949     952
950     953
951     954
952     955
953     956
954     957
955     958
956     959
957     960
958     961
959     962
960     963
961     964
962     965
963     966
964     967
965     968
966     969
967     970
968     971
969     972
970     973
971     974
972     975
973     976
974     977
975     978
976     979
977     980
978     981
979     982
980     983
981     984
982     985
983     986
984     987
985     988
986     989
987     990
988     991
989     992
990     993
991     994
992     995
993     996
994     997
995     998
996     999
997     1000
998     1001
999     1002
1000     1003
1001     1004
1002     1005
1003     1006
1004     1007
1005     1008
1006     1009
1007     1010
1008     1011
1009     1012
1010     1013
1011     1014
1012     1015
1013     1016
1014     1017
1015     1018
1016     1019
1017     1020
1018     1021
1019     1022
1020     1023
1021     1024
1022     1025
1023     1026
1024     1027
1025     1028
1026     1029
1027     1030
1028     1031
1029     1032
1030     1033
1031     1034
1032     1035
1033     1036
1034     1037
1035     1038
1036     1039
1037     1040
1038     1041
1039     1042
1040     1043
1041     1044
1042     1045
1043     1046
1044     1047
1045     1048
1046     1049
1047     1050
1048     1051
1049     1052
1050     1053
1051     1054
1052     1055
1053     1056
1054     1057
1055     1058
1056     1059
1057     1060
1058     1061
1059     1062
1060     1063
1061     1064
1062     1065
1063     1066
1064     1067
1065     1068
1066     1069
1067     1070
1068     1071
1069     1072
1070     1073
1071     1074
1072     1075
1073     1076
1074     1077
1075     1078
1076     1079
1077     1080
1078     1081
1079     1082
1080     1083
1081     1084
1082     1085
1083     1086
1084     1087
1085     1088
1086     1089
1087     1090
1088     1091
1089     1092
1090     1093
1091     1094
1092     1095
1093     1096
1094     1097
1095     1098
1096     1099
1097     1100
1098     1101
1099     1102
1100     1103
1101     1104
1102     1105
1103     1106
1104     1107
1105     1108
1106     1109
1107     1110
1108     1111
1109     1112
1110     1113
1111     1114
1112     1115
1113     1116
1114     1117
1115     1118
1116     1119
1117     1120
1118     1121
1119     1122
1120     1123
1121     1124
1122     1125
1123     1126
1124     1127
1125     1128
1126     1129
1127     1130
1128     1131
1129     1132
1130     1133
1131     1134
1132     1135
1133     1136
1134     1137
1135     1138
1136     1139
1137     1140
1138     1141
1139     1142
1140     1143
1141     1144
1142     1145
1143     1146
1144     1147
1145     1148
1146     1149
1147     1150
1148     1151
1149     1152
1150     1153
1151     1154
1152     1155
1153     1156
1154     1157
1155     1158
1156     1159
1157     1160
1158     1161
1159     1162
1160     1163
1161     1164
1162     1165
1163     1166
1164     1167
1165     1168
1166     1169
1167     1170
1168     1171
1169     1172
1170     1173
1171     1174
1172     1175
1173     1176
1174     1177
1175     1178
1176     1179
1177     1180
1178     1181
1179     1182
1180     1183
1181     1184
1182     1185
1183     1186
1184     1187
1185     1188
1186     1189
1187     1190
1188     1191
1189     1192
1190     1193
1191     1194
1192     1195
1193     1196
1194     1197
1195     1198
1196     1199
1197     1200
1198     1201
1199     1202
1200     1203
1201     1204
1202     1205
1203     1206
1204     1207
1205     1208
1206     1209
1207     1210
1208     1211
1209     1212
1210     1213
1211     1214
1212     1215
1213     1216
1214     1217
1215     1218
1216     1219
1217     1220
1218     1221
1219     1222
1220     1223
1221     1224
1222     1225
1223     1226
1224     1227
1225     1228
1226     1229
1227     1230
1228     1231
1229     1232
1230     1233
1231     1234
1232     1235
1233     1236
1234     1237
1235     1238
1236     1239
1237     1240
1238     1241
1239     1242
1240     1243
1241     1244
1242     1245
1243     1246
1244     1247
1245     1248
1246     1249
1247     1250
1248     1251
1249     1252
1250     1253
1251     1254
1252     1255
1253     1256
1254     1257
1255     1258
1256     1259
1257     1260
1258     1261
1259     1262
1260     1263
1261     1264
1262     1265
1263     1266
1264     1267
1265     1268
1266     1269
1267     1270
1268     1271
1269     1272
1270     1273
1271     1274
1272     1275
1273     1276
1274     1277
1275     1278
1276     1279
1277     1280
1278     1281
1279     1282
1280     1283
1281     1284
1282     1285
1283     1286
1284     1287
1285     1288
1286     1289
1287     1290
1288     1291
1289     1292
1290     1293
1291     1294
1292     1295
1293     1296
1294     1297
1295     1298
1296     1299
1297     1300
1298     1301
1299     1302
1300     1303
1301     1304
1302     1305
1303     1306
1304     1307
1305     1308
1306     1309
1307     1310
1308     1311
1309     1312
1310     1313
1311     1314
1312     1315
1313     1316
1314     1317
1315     1318
1316     1319
1317     1320
1318     1321
1319     1322
1320     1323
1321     1324
1322     1325
1323     1326
1324     1327
1325     1328
1326     1329
1327     1330
```

```

111 -         'foo/bar.txt'
112 -     >>> san('foo../.bar.txt')
113 -         'foo../.bar.txt'
114 -     >>> san('\\\\foo\\bar.txt')
115 -         'foo/bar.txt'
116 -     >>> san('D:\\foo.txt')
117 -         'D/foo.txt'
118 -     >>> san('\\\\\\server\\share\\file.txt')
119 -         'server/share/file.txt'
120 -     >>> san('\\\\\\?\\GLOBALROOT\\Volume3')
121 -         '?/GLOBALROOT/Volume3'
122 -     >>> san('\\\\\\.\PhysicalDrive1\\root')
123 -         'PhysicalDrive1/root'
124 -
125 -     Retain any trailing slash.
126 -     >>> san('abc/')
127 -         'abc/'
128 -
129 -     Raises a ValueError if the result is
empty.
130 -     >>> san('../..')
131 -     Traceback (most recent call last):
132 -     ...
133 -     ValueError: Empty filename
134 -     """
135 -
136 -     def allowed(part):
137 -         return part and part not in {'..',
'.'}
138 -
139 -     # Remove the drive letter.
140 -     # Don't use ntpath.splitdrive, because
that also strips UNC paths
141 -     bare = re.sub('^([A-Z]):', r'\1', name,
flags=re.IGNORECASE)
142 -     clean = bare.replace('\\\\', '/')
143 -     parts = clean.split('/')
144 -     joined = '/'.join(filter(allowed,
parts))
145 -     if not joined:
146 -         raise ValueError("Empty filename")
147 -     return joined + '/' *
name.endswith('/')
148 -
149 -
150 - class CompleteDirs(InitializedState,
SanitizedNames, zipfile.ZipFile):
102 + class CompleteDirs(InitializedState,
zipfile.ZipFile):
151 103     """
152 104     A ZipFile subclass that ensures that
implied directories
153 105     are always included in the namelist.

```

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

0 comments on commit 2231286

Please [sign in](#) to comment.