

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

```

3025 +         assert list(map(str, root.iterdir()))
3026 +         == ['../']
3027 +
3028 +         assert
3029 +         root.joinpath('..').joinpath('parent.txt').read
3030 +         _bytes() == b'content'
3031 +
3032 +         def test_unsupported_names(self):
3033 +             """
3034 +             Path segments with special characters
3035 +             are readable.
3036 +
3037 +             On some platforms or file systems,
3038 +             characters like
3039 +             ``:`` and ``?`` are not allowed, but
3040 +             they are valid
3041 +             in the zip file.
3042 +             """
3043 +             data = io.BytesIO()
3044 +             zf = zipfile.ZipFile(data, "w")
3045 +             zf.writestr("path?", b"content")
3046 +             zf.writestr("V: NMS.flac", b"fLaC...")
3047 +             zf.filename = ''
3048 +             root = zipfile.Path(zf)
3049 +             contents = root.iterdir()
3050 +             assert next(contents).name == 'path?'
3051 +             assert next(contents).name == 'V:
3052 +             NMS.flac'
3053 +             assert root.joinpath('V:
3054 +             NMS.flac').read_bytes() == b"fLaC..."
3055 +
3056 +         def test_backslash_not_separator(self):
3057 +             """
3058 +             In a zip file, backslashes are not
3059 +             separators.
3060 +             """
3061 +             data = io.BytesIO()
3062 +             zf = zipfile.ZipFile(data, "w")
3063 +             zf.writestr(DirtyZipInfo.for_name("foo\\bar",
3064 +             zf), b"content")
3065 +             zf.filename = ''
3066 +             root = zipfile.Path(zf)
3067 +             (first,) = root.iterdir()
3068 +             assert not first.is_dir()
3069 +             assert first.name == 'foo\\bar'
3070 +
3071 +         class DirtyZipInfo(zipfile.ZipInfo):
3072 +             """
3073 +             Bypass name sanitization.
3074 +             """
3075 +
3076 +             def __init__(self, filename, *args,
3077 +             **kwargs):
3078 +                 super().__init__(filename, *args,
3079 +                 **kwargs)

```

```

3068 +         self.filename = filename
3069 +
3070 +     @classmethod
3071 +     def for_name(cls, name, archive):
3072 +         """
3073 +         Construct the same way that
3074 +         ZipFile.writestr does.
3075 +
3076 +         TODO: extract this functionality and
3077 +         re-use
3078 +         """
3079 +         self = cls(filename=name,
3080 +                     date_time=time.localtime(time.time())[:6])
3081 +         self.compress_type =
3082 +         archive.compression
3083 +         self.compress_level =
3084 +         archive.compresslevel
3085 +         if self.filename.endswith('/'): #
3086 +             pragma: no cover
3087 +             self.external_attr = 0o40775 << 16
3088 +             # drwxrwxr-x
3089 +             self.external_attr |= 0x10 # MS-
3090 +             DOS directory flag
3091 +         else:
3092 +             self.external_attr = 0o600 << 16 #
3093 +             ?rw-----
3094 +         return self
3095 +
3096 +
3097 +
3098 +
3099 +
3100 +
3101 +
3102 +
3103 +
3104 +
3105 +
3106 +
3107 +
3108 +
3109 +
3110 +
3111 +
3112 +
3113 +
3114 +
3115 +
3116 +
3117 +
3118 +
3119 +
3120 +
3121 +
3122 +
3123 +
3124 +
3125 +
3126 +
3127 +
3128 +
3129 +
3130 +
3131 +
3132 +
3133 +
3134 +
3135 +
3136 +
3137 +
3138 +
3139 +
3140 +
3141 +
3142 +
3143 +
3144 +
3145 +
3146 +
3147 +
3148 +
3149 +
3150 +
3151 +
3152 +
3153 +
3154 +
3155 +
3156 +
3157 +
3158 +
3159 +
3160 +
3161 +
3162 +
3163 +
3164 +
3165 +
3166 +
3167 +
3168 +
3169 +
3170 +
3171 +
3172 +
3173 +
3174 +
3175 +
3176 +
3177 +
3178 +
3179 +
3180 +
3181 +
3182 +
3183 +
3184 +
3185 +
3186 +
3187 +
3188 +
3189 +
3190 +
3191 +
3192 +
3193 +
3194 +
3195 +
3196 +
3197 +
3198 +
3199 +
3200 +
3201 +
3202 +
3203 +
3204 +
3205 +
3206 +
3207 +
3208 +
3209 +
3210 +
3211 +
3212 +
3213 +
3214 +
3215 +
3216 +
3217 +
3218 +
3219 +
3220 +
3221 +
3222 +
3223 +
3224 +
3225 +
3226 +
3227 +
3228 +
3229 +
3230 +
3231 +
3232 +
3233 +
3234 +
3235 +
3236 +
3237 +
3238 +
3239 +
3240 +
3241 +
3242 +
3243 +
3244 +
3245 +
3246 +
3247 +
3248 +
3249 +
3250 +
3251 +
3252 +
3253 +
3254 +
3255 +
3256 +
3257 +
3258 +
3259 +
3260 +
3261 +
3262 +
3263 +
3264 +
3265 +
3266 +
3267 +
3268 +
3269 +
3270 +
3271 +
3272 +
3273 +
3274 +
3275 +
3276 +
3277 +
3278 +
3279 +
3280 +
3281 +
3282 +
3283 +
3284 +
3285 +
3286 +
3287 +
3288 +
3289 +
3290 +
3291 +
3292 +
3293 +
3294 +
3295 +
3296 +
3297 +
3298 +
3299 +
3300 +
3301 +
3302 +
3303 +
3304 +
3305 +
3306 +
3307 +
3308 +
3309 +
3310 +
3311 +
3312 +
3313 +
3314 +
3315 +
3316 +
3317 +
3318 +
3319 +
3320 +
3321 +
3322 +
3323 +
3324 +
3325 +
3326 +
3327 +
3328 +
3329 +
3330 +
3331 +
3332 +
3333 +
3334 +
3335 +
3336 +
3337 +
3338 +
3339 +
3340 +
3341 +
3342 +
3343 +
3344 +
3345 +
3346 +
3347 +
3348 +
3349 +
3350 +
3351 +
3352 +
3353 +
3354 +
3355 +
3356 +
3357 +
3358 +
3359 +
3360 +
3361 +
3362 +
3363 +
3364 +
3365 +
3366 +
3367 +
3368 +
3369 +
3370 +
3371 +
3372 +
3373 +
3374 +
3375 +
3376 +
3377 +
3378 +
3379 +
3380 +
3381 +
3382 +
3383 +
3384 +
3385 +
3386 +
3387 +
3388 +
3389 +
3390 +
3391 +
3392 +
3393 +
3394 +
3395 +
3396 +
3397 +
3398 +
3399 +
3400 +
3401 +
3402 +
3403 +
3404 +
3405 +
3406 +
3407 +
3408 +
3409 +
3410 +
3411 +
3412 +
3413 +
3414 +
3415 +
3416 +
3417 +
3418 +
3419 +
3420 +
3421 +
3422 +
3423 +
3424 +
3425 +
3426 +
3427 +
3428 +
3429 +
3430 +
3431 +
3432 +
3433 +
3434 +
3435 +
3436 +
3437 +
3438 +
3439 +
3440 +
3441 +
3442 +
3443 +
3444 +
3445 +
3446 +
3447 +
3448 +
3449 +
3450 +
3451 +
3452 +
3453 +
3454 +
3455 +
3456 +
3457 +
3458 +
3459 +
3460 +
3461 +
3462 +
3463 +
3464 +
3465 +
3466 +
3467 +
3468 +
3469 +
3470 +
3471 +
3472 +
3473 +
3474 +
3475 +
3476 +
3477 +
3478 +
3479 +
3480 +
3481 +
3482 +
3483 +
3484 +
3485 +
3486 +
3487 +
3488 +
3489 +
3490 +
3491 +
3492 +
3493 +
3494 +
3495 +
3496 +
3497 +
3498 +
3499 +
3500 +
3501 +
3502 +
3503 +
3504 +
3505 +
3506 +
3507 +
3508 +
3509 +
3510 +
3511 +
3512 +
3513 +
3514 +
3515 +
3516 +
3517 +
3518 +
3519 +
3520 +
3521 +
3522 +
3523 +
3524 +
3525 +
3526 +
3527 +
3528 +
3529 +
3530 +
3531 +
3532 +
3533 +
3534 +
3535 +
3536 +
3537 +
3538 +
3539 +
3540 +
3541 +
3542 +
3543 +
3544 +
3545 +
3546 +
3547 +
3548 +
3549 +
3550 +
3551 +
3552 +
3553 +
3554 +
3555 +
3556 +
3557 +
3558 +
3559 +
3560 +
3561 +
3562 +
3563 +
3564 +
3565 +
3566 +
3567 +
3568 +
3569 +
3570 +
3571 +
3572 +
3573 +
3574 +
3575 +
3576 +
3577 +
3578 +
3579 +
3580 +
3581 +
3582 +
3583 +
3584 +
3585 +
3586 +
3587 +
3588 +
3589 +
3590 +
3591 +
3592 +
3593 +
3594 +
3595 +
3596 +
3597 +
3598 +
3599 +
3600 +
3601 +
3602 +
3603 +
3604 +
3605 +
3606 +
3607 +
3608 +
3609 +
3610 +
3611 +
3612 +
3613 +
3614 +
3615 +
3616 +
3617 +
3618 +
3619 +
3620 +
3621 +
3622 +
3623 +
3624 +
3625 +
3626 +
3627 +
3628 +
3629 +
3630 +
3631 +
3632 +
3633 +
3634 +
3635 +
3636 +
3637 +
3638 +
3639 +
3640 +
3641 +
3642 +
3643 +
3644 +
3645 +
3646 +
3647 +
3648 +
3649 +
3650 +
3651 +
3652 +
3653 +
3654 +
3655 +
3656 +
3657 +
3658 +
3659 +
3660 +
3661 +
3662 +
3663 +
3664 +
3665 +
3666 +
3667 +
3668 +
3669 +
3670 +
3671 +
3672 +
3673 +
3674 +
3675 +
3676 +
3677 +
3678 +
3679 +
3680 +
3681 +
3682 +
3683 +
3684 +
3685 +
3686 +
3687 +
3688 +
3689 +
3690 +
3691 +
3692 +
3693 +
3694 +
3695 +
3696 +
3697 +
3698 +
3699 +
3700 +
3701 +
3702 +
3703 +
3704 +
3705 +
3706 +
3707 +
3708 +
3709 +
3710 +
3711 +
3712 +
3713 +
3714 +
3715 +
3716 +
3717 +
3718 +
3719 +
3720 +
3721 +
```

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']
2181 """
2177 2182 path = path.rstrip(posixpath.sep)
2178 - while path and path != posixpath.sep:
2183 + while path.rstrip(posixpath.sep):

```

2179	2184	<code>yield path</code>
2180	2185	<code>path, tail = posixpath.split(path)</code>
2181	2186	

▼ 3    

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

...	...	<code>@@ -0,0 +1,3 @@</code>
	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 7bc367e

Please [sign in](#) to comment.