

python / cpython Public

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

Commit 9c1110e



6 people authored on Jun 3, 2025 · ✖ 24 / 26 · Partially verified

```
[3.10] gh-135034: Normalize link targets in tarfile, add
os.path.realpath(strict='allow_missing') (GH-135037) (#135070)

Addresses CVEs 2024-12718, 2025-4138, 2025-4330, and 2025-4517.
(cherry picked from commit 3612d8f)
(cherry picked from commit c358142)
(cherry picked from commit 371b4ea)

Co-authored-by: Łukasz Langa <lukasz@langa.pl>
Signed-off-by: Łukasz Langa <lukasz@langa.pl>
Co-authored-by: Petr Viktorin <encukou@gmail.com>
Co-authored-by: Seth Michael Larson <seth@python.org>
Co-authored-by: Adam Turner <9087854+AA-Turner@users.noreply.github.com>
Co-authored-by: Serhiy Storchaka <storchaka@gmail.com>
```

3.10 (#135070) · v3.10.20 ... v3.10.18

1 parent c71ea4b commit 9c1110e

11 files changed +944 -134 lines changed

↑ Top ⚙️

Filter files...

- ▼ Doc
 - ▼ library
 - os.path.rst
 - tarfile.rst
 - ▼ whatsnew
 - 3.10.rst
 - ▼ Lib
 - genericpath.py
 - ntpath.py

- 📄 posixpath.py
- 📄 tarfile.py
- ▼ 📁 test
 - 📄 test_ntpath.py
 - 📄 test_posixpath.py
 - 📄 test_tarfile.py
- ▼ 📁 Misc/NEWS.d/next/Security
 - 📄 2025-06-02-11-32-23.gh-issue-135034.RLGjbp.rst

📄 **11 files changed** +944 -134 lines changed

🔍 Search within code



▼ Doc/library/os.path.rst
⏪ 📄 ⋮

↑	@@ -352,10 +352,26 @@ the :mod:`glob` module.)
352	352 links encountered in the path (if they are supported by the operating
353	353 system).
354	354
355	- If a path doesn't exist or a symlink loop is encountered, and <i>strict</i> is
356	- ``True``, <code>:exc:`OSError`</code> is raised. If <i>strict</i> is ``False``, the path is
357	- resolved as far as possible and any remainder is appended without checking
358	- whether it exists.
355	+ By default, the path is evaluated up to the first component that does not
356	+ exist, is a symlink loop, or whose evaluation raises <code>:exc:`OSError`</code> .
357	+ All such components are appended unchanged to the existing part of the path.
358	+
359	+ Some errors that are handled this way include "access denied", "not a
360	+ directory", or "bad argument to internal function". Thus, the
361	+ resulting path may be missing or inaccessible, may still contain
362	+ links or loops, and may traverse non-directories.
363	+
364	+ This behavior can be modified by keyword arguments:
365	+
366	+ If <i>strict</i> is ``True``, the first error encountered when evaluating the
367	+ path is
368	+ re-raised.
369	+ In particular, <code>:exc:`FileNotFoundError`</code> is raised if <i>path</i> does not exist,
370	+ or another <code>:exc:`OSError`</code> if it is otherwise inaccessible.

```

371 + If *strict* is :py:data:`os.path.ALLOW_MISSING`, errors other than
372 + :exc:`FileNotFoundError` are re-raised (as with ``strict=True``).
373 + Thus, the returned path will not contain any symbolic links, but the named
374 + file and some of its parent directories may be missing.

359 375
360 376     .. note::
361 377         This function emulates the operating system's procedure for making a path
@@ -374,6 +390,15 @@ the :mod:`glob` module.)
374 390     .. versionchanged:: 3.10
375 391         The *strict* parameter was added.
376 392

393 +     .. versionchanged:: next
394 +         The :py:data:`~os.path.ALLOW_MISSING` value for the *strict* parameter
395 +         was added.
396 +
397 + .. data:: ALLOW_MISSING
398 +
399 +     Special value used for the *strict* argument in :func:`realpath`.
400 +
401 +     .. versionadded:: next

377 402
378 403     .. function:: relpath(path, start=os.curdir)
379 404

```

Doc/library/tarfile.rst

```

@@ -237,6 +237,15 @@ The :mod:`tarfile` module defines the following
exceptions:

237 237     Raised to refuse extracting a symbolic link pointing outside the
destination
238 238     directory.
239 239

240 + .. exception:: LinkFallbackError
241 +
242 +     Raised to refuse emulating a link (hard or symbolic) by extracting another
243 +     archive member, when that member would be rejected by the filter location.
244 +     The exception that was raised to reject the replacement member is
available
245 +     as :attr:`!BaseException.__context__`.
246 +

```

247	+	.. <code>versionadded:: next</code>
248	+	
240	249	
241	250	The following constants are available at the module level:
242	251	
⌵		@@ -954,6 +963,12 @@ reused in custom filters:
954	963	Implements the ``'data'`` filter.
955	964	In addition to what ``tar_filter`` does:
956	965	
966	+	- Normalize link targets (:attr:`TarInfo.linkname`) using
967	+	:func:`os.path.normpath`.
968	+	Note that this removes internal ``..`` components, which may change the
969	+	meaning of the link if the path in :attr:`!TarInfo.linkname` traverses
970	+	symbolic links.
971	+	
957	972	- :ref:`Refuse <tarfile-extraction-refuse>` to extract links (hard or soft)
958	973	that link to absolute paths, or ones that link outside the destination.
959	974	
⌵		@@ -982,6 +997,10 @@ reused in custom filters:
982	997	
983	998	Return the modified ``TarInfo`` member.
984	999	
1000	+	.. <code>versionchanged:: next</code>
1001	+	
1002	+	Link targets are now normalized.
1003	+	
985	1004	
986	1005	.. <code>_tarfile-extraction-refuse:</code>
987	1006	
⬆		@@ -1008,6 +1027,7 @@ Here is an incomplete list of things to consider:
1008	1027	* Extract to a :func:`new temporary directory <tempfile.mkdtemp>`
1009	1028	to prevent e.g. exploiting pre-existing links, and to make it easier to
1010	1029	clean up after a failed extraction.
1030	+	* Disallow symbolic links if you do not need the functionality.
1011	1031	* When working with untrusted data, use external (e.g. OS-level) limits on
1012	1032	disk, memory and CPU usage.
1013	1033	* Check filenames against an allow-list of characters
⌵		

Doc/whatsnew/3.10.rst



↑

@@ -2394,3 +2394,37 @@ email

2394 2394 check `if` the `*strict*` parameter `is` available.
 2395 2395 (Contributed by Thomas Dwyer and Victor Stinner for :gh:`102988` to improve
 2396 2396 the [CVE-2023-27043](#) fix.)

```

2397 +
2398 +
2399 + Notable changes in 3.10.18
2400 + =====
2401 +
2402 + os.path
2403 + -----
2404 +
2405 + * The *strict* parameter to :func:`os.path.realpath` accepts a new value,
2406 + :data:`os.path.ALLOW_MISSING`.
2407 + If used, errors other than :exc:`FileNotFoundError` will be re-raised;
2408 + the resulting path can be missing but it will be free of symlinks.
2409 + (Contributed by Petr Viktorin for CVE 2025-4517.)
2410 +
2411 + tarfile
2412 + -----
2413 +
2414 + * :func:`~tarfile.data_filter` now normalizes symbolic link targets in order
2415 + to
2416 + avoid path traversal attacks.
2417 + (Contributed by Petr Viktorin in :gh:`127987` and CVE 2025-4138.)
2418 + * :func:`~tarfile.TarFile.extractall` now skips fixing up directory
2419 + attributes
2420 + when a directory was removed or replaced by another kind of file.
2421 + (Contributed by Petr Viktorin in :gh:`127987` and CVE 2024-12718.)
2422 + * :func:`~tarfile.TarFile.extract` and :func:`~tarfile.TarFile.extractall`
2423 + now (re-)apply the extraction filter when substituting a link (hard or
2424 + symbolic) with a copy of another archive member, and when fixing up
2425 + directory attributes.
2426 + The former raises a new exception, :exc:`~tarfile.LinkFallbackError`.
2427 + (Contributed by Petr Viktorin for CVE 2025-4330 and CVE 2024-12718.)
2428 + * :func:`~tarfile.TarFile.extract` and :func:`~tarfile.TarFile.extractall`
2429 + no longer extract rejected members when
2430 + :func:`~tarfile.TarFile.errorlevel` is zero.
```

2429 + (Contributed by Matt Prodan and Petr Viktorin in :gh:`112887`
 2430 + and CVE 2025-4435.)

Lib/genericpath.py

↑

@@ -8,7 +8,7 @@

```

8      8
9      9     __all__ = ['commonprefix', 'exists', 'getatime', 'getctime', 'getmtime',
10     10         'getsize', 'isdir', 'isfile', 'samefile', 'sameopenfile',
11     -         'samestat']
11     +         'samestat', 'ALLOW_MISSING']

```

```

12     12
13     13
14     14     # Does a path exist?

```

↓

↑

@@ -153,3 +153,12 @@ def _check_arg_types(funcname, *args):

```

153     153         f'os.PathLike object, not
        {s.__class__.__name__!r}') from None
154     154         if hasstr and hasbytes:
155     155             raise TypeError("Can't mix strings and bytes in path components") from
        None
156     +
157     + # A singleton with a true boolean value.
158     + @object.__new__
159     + class ALLOW_MISSING:
160     +     """Special value for use in realpath()."""
161     +     def __repr__(self):
162     +         return 'os.path.ALLOW_MISSING'
163     +     def __reduce__(self):
164     +         return self.__class__.__name__

```

Lib/ntpath.py

↑

@@ -30,7 +30,8 @@

```

30     30         "ismount", "expanduser", "expandvars", "normpath", "abspath",
31     31         "curdir", "pardir", "sep", "pathsep", "defpath", "altsep",
32     32
        "extsep", "devnull", "realpath", "supports_unicode_filenames", "relpath",
33     -         "samefile", "sameopenfile", "samestat", "commonpath"]
33     +         "samefile", "sameopenfile", "samestat", "commonpath",
34     +         "ALLOW_MISSING"]

```

```

34 35
35 36     def _get_bothseps(path):
36 37         if isinstance(path, bytes):
37 38             return path
38 39
39 40     @@ -571,9 +572,10 @@ def abspath(path):
40 41
41 42         from nt import _getfinalpathname, readlink as _nt_readlink
42 43     except ImportError:
43 44         # realpath is a no-op on systems without _getfinalpathname support.
44 45     -   realpath = abspath
45 46     +   def realpath(path, *, strict=False):
46 47         +       return abspath(path)
47 48
48 49     else:
49 50     -   def _readlink_deep(path):
50 51     +   def _readlink_deep(path, ignored_error=OSError):
51 52         # These error codes indicate that we should stop reading links and
52 53         # return the path we currently have.
53 54         # 1: ERROR_INVALID_FUNCTION
54 55
55 56     @@ -606,7 +608,7 @@ def _readlink_deep(path):
56 57
57 58         path = old_path
58 59         break
59 60         path = normpath(join(dirname(old_path), path))
60 61     -   except OSError as ex:
61 62     +   except ignored_error as ex:
62 63         if ex.winerror in allowed_winerror:
63 64             break
64 65         raise
65 66
66 67     @@ -615,7 +617,7 @@ def _readlink_deep(path):
67 68         break
68 69         return path
69 70
70 71     -   def _getfinalpathname_nonstrict(path):
71 72     +   def _getfinalpathname_nonstrict(path, ignored_error=OSError):
72 73         # These error codes indicate that we should stop resolving the path
73 74         # and return the value we currently have.
74 75         # 1: ERROR_INVALID_FUNCTION
75 76
76 77     @@ -642,17 +644,18 @@ def _getfinalpathname_nonstrict(path):
77 78         try:
78 79             path = _getfinalpathname(path)
79 80         return join(path, tail) if tail else path

```

```

645 -         except OSError as ex:
647 +         except ignored_error as ex:
646 648             if ex.winerror not in allowed_winerror:
647 649                 raise
648 650             try:
649 651                 # The OS could not resolve this path fully, so we attempt
650 652                 # to follow the link ourselves. If we succeed, join the
651 653                 tail
651 653                 # and return.
652 -                 new_path = _readlink_deep(path)
654 +                 new_path = _readlink_deep(path,
655 +                                         ignored_error=ignored_error)
653 656                 if new_path != path:
654 657                     return join(new_path, tail) if tail else new_path
655 -         except OSError:
658 +         except ignored_error:
656 659             # If we fail to readlink(), let's keep traversing
657 660             pass
658 661             path, name = split(path)
659 -         @@ -683,16 +686,24 @@ def realpath(path, *, strict=False):
683 686             if normcase(path) == normcase(devnull):
684 687                 return '\\\\.\\NUL'
685 688             had_prefix = path.startswith(prefix)
689 +
690 +         if strict is ALLOW_MISSING:
691 +             ignored_error = FileNotFoundError
692 +             strict = True
693 +         elif strict:
694 +             ignored_error = ()
695 +         else:
696 +             ignored_error = OSError
697 +
686 698             if not had_prefix and not isabs(path):
687 699                 path = join(cwd, path)
688 700             try:
689 701                 path = _getfinalpathname(path)
690 702                 initial_winerror = 0
691 -         except OSError as ex:
692 -             if strict:

```

```

693 - raise
694 703 + except ignored_error as ex:
694 704     initial_winerror = ex.winerror
695 - path = _getfinalpathname_nonstrict(path)
696 705 + path = _getfinalpathname_nonstrict(path,
697 706 +     ignored_error=ignored_error)
698 707     # The path returned by _getfinalpathname will always start with \\?\ -
699 708     # strip off that prefix unless it was already provided on the original
700 709     # path.

```

Lib/posixpath.py

```

@@ -35,7 +35,7 @@
35 35     "samefile", "sameopenfile", "samestat",
36 36     "curdir", "pardir", "sep", "pathsep", "defpath", "altsep", "extsep",
37 37     "devnull", "realpath", "supports_unicode_filenames", "relpath",
38 -     "commonpath"]
38 +     "commonpath", "ALLOW_MISSING"]
39 39
40 40
41 41 def _get_sep(path):
@@ -407,6 +407,15 @@ def _joinrealpath(path, rest, strict, seen):
407 407     sep = '/'
408 408     curdir = '.'
409 409     pardir = '..'
410 +     getcwd = os.getcwd
411 +     if strict is ALLOW_MISSING:
412 +         ignored_error = FileNotFoundError
413 +     elif strict:
414 +         ignored_error = ()
415 +     else:
416 +         ignored_error = OSError
417 +
418 +     maxlinks = None
419 419
420 420     if isabs(rest):
421 421         rest = rest[1:]
@@ -429,9 +438,7 @@ def _joinrealpath(path, rest, strict, seen):
429 438     newpath = join(path, name)

```

```
430 439         try:
431 440             st = os.lstat(newpath)
432 -         except OSError:
433 -             if strict:
434 -                 raise
441 +         except ignored_error:
435 442             is_link = False
436 443         else:
437 444             is_link = stat.S_ISLNK(st.st_mode)
```



Comments 0