python / **cpython** Public

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

Commit 19de092

6 people authored on Jun 3, 2025 · ✕ 22 / 25 · Partially verified

```
[3.12] gh-135034: Normalize link targets in tarfile, add
os.path.realpath(strict='allow_missing') (GH-135037) (GH-135066)

Addresses CVEs 2024-12718, 2025-4138, 2025-4330, and 2025-4517.

(cherry picked from commit 3612d8f)

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>
```

3.12 (#135066) · v3.12.13 ••• v3.12.11

1 parent f3272d8 commit 19de092

11 files changed +1,064 -139

Top

Filter files…

- Doc
  - library
    - os.path.rst
    - tarfile.rst
  - whatsnew
    - 3.12.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

Search within code

Doc/library/os.path.rst

```
@@ -377,10 +377,26 @@ the :mod:`glob` module.)
377 377     links encountered in the path (if they are supported by the operating
378 378     system).
379 379
380     -   If a path doesn't exist or a symlink loop is encountered, and *strict* is
381     -   ``True``, :exc:`OSError` is raised. If *strict* is ``False``, the path is
382     -   resolved as far as possible and any remainder is appended without checking
383     -   whether it exists.
380 +   By default, the path is evaluated up to the first component that does not
381 +   exist, is a symlink loop, or whose evaluation raises :exc:`OSError`.
382 +   All such components are appended unchanged to the existing part of the path.
383 +
384 +   Some errors that are handled this way include "access denied", "not a
385 +   directory", or "bad argument to internal function". Thus, the
386 +   resulting path may be missing or inaccessible, may still contain
387 +   links or loops, and may traverse non-directories.
388 +
389 +   This behavior can be modified by keyword arguments:
390 +
391 +   If *strict* is ``True``, the first error encountered when evaluating the
    path is
392 +   re-raised.
393 +   In particular, :exc:`FileNotFoundError` is raised if *path* does not exist,
```

```
                394 +    or another :exc:`OSError` if it is otherwise inaccessible.
                395 +
                396 +    If *strict* is :py:data:`os.path.ALLOW_MISSING`, errors other than
                397 +    :exc:`FileNotFoundError` are re-raised (as with ``strict=True``).
                398 +    Thus, the returned path will not contain any symbolic links, but the named
                399 +    file and some of its parent directories may be missing.
384             400
385             401        .. note::
386             402           This function emulates the operating system's procedure for making a path
                   @@ -399,6 +415,15 @@ the :mod:`glob` module.)
399             415        .. versionchanged:: 3.10
400             416           The *strict* parameter was added.
401             417
                418 +    .. versionchanged:: next
                419 +       The :py:data:`~os.path.ALLOW_MISSING` value for the *strict* parameter
                420 +       was added.
                421 +
                422 + .. data:: ALLOW_MISSING
                423 +
                424 +    Special value used for the *strict* argument in :func:`realpath`.
                425 +
                426 +    .. versionadded:: next
402             427
403             428    .. function:: relpath(path, start=os.curdir)
404             429
```

Doc/library/tarfile.rst

```
                   @@ -249,6 +249,15 @@ The :mod:`tarfile` module defines the following
                   exceptions:
249             249        Raised to refuse extracting a symbolic link pointing outside the
                   destination
250             250        directory.
251             251
                252 + .. exception:: LinkFallbackError
                253 +
                254 +    Raised to refuse emulating a link (hard or symbolic) by extracting another
                255 +    archive member, when that member would be rejected by the filter location.
                256 +    The exception that was raised to reject the replacement member is
                   available
```

```
                257 +    as :attr:`!BaseException.__context__`.
                258 +
                259 +    .. versionadded:: next
                260 +
252             261
253             262     The following constants are available at the module level:
254             263
                        @@ -1039,6 +1048,12 @@ reused in custom filters:
1039            1048      Implements the ``'data'`` filter.
1040            1049      In addition to what ``tar_filter`` does:
1041            1050
                1051 +    - Normalize link targets (:attr:`TarInfo.linkname`) using
                1052 +      :func:`os.path.normpath`.
                1053 +      Note that this removes internal ``..`` components, which may change the
                1054 +      meaning of the link if the path in :attr:`!TarInfo.linkname` traverses
                1055 +      symbolic links.
                1056 +
1042            1057      - :ref:`Refuse <tarfile-extraction-refuse>` to extract links (hard or soft)
1043            1058        that link to absolute paths, or ones that link outside the destination.
1044            1059
                        @@ -1067,6 +1082,10 @@ reused in custom filters:
1067            1082
1068            1083      Return the modified ``TarInfo`` member.
1069            1084
                1085 +    .. versionchanged:: next
                1086 +
                1087 +       Link targets are now normalized.
                1088 +
1070            1089
1071            1090    .. _tarfile-extraction-refuse:
1072            1091
                        @@ -1093,6 +1112,7 @@ Here is an incomplete list of things to consider:
1093            1112    * Extract to a :func:`new temporary directory <tempfile.mkdtemp>`
1094            1113      to prevent e.g. exploiting pre-existing links, and to make it easier to
1095            1114      clean up after a failed extraction.
                1115 + * Disallow symbolic links if you do not need the functionality.
1096            1116    * When working with untrusted data, use external (e.g. OS-level) limits on
1097            1117      disk, memory and CPU usage.
```

```
1098  1118    * Check filenames against an allow-list of characters
```

Doc/whatsnew/3.12.rst

```
      @@ -2320,3 +2320,37 @@ sys
2320  2320    * The previously undocumented special function :func:`sys.getobjects`,
2321  2321      which only exists in specialized builds of Python, may now return objects
2322  2322      from other interpreters than the one it's called in.
      2323  +
      2324  +
      2325  + Notable changes in 3.12.10
      2326  + ==========================
      2327  +
      2328  + os.path
      2329  + -------
      2330  +
      2331  + * The *strict* parameter to :func:`os.path.realpath` accepts a new value,
      2332  +   :data:`os.path.ALLOW_MISSING`.
      2333  +   If used, errors other than :exc:`FileNotFoundError` will be re-raised;
      2334  +   the resulting path can be missing but it will be free of symlinks.
      2335  +   (Contributed by Petr Viktorin for :cve:`2025-4517`.)
      2336  +
      2337  + tarfile
      2338  + -------
      2339  +
      2340  + * :func:`~tarfile.data_filter` now normalizes symbolic link targets in order
      to
      2341  +   avoid path traversal attacks.
      2342  +   (Contributed by Petr Viktorin in :gh:`127987` and :cve:`2025-4138`.)
      2343  + * :func:`~tarfile.TarFile.extractall` now skips fixing up directory
      attributes
      2344  +   when a directory was removed or replaced by another kind of file.
      2345  +   (Contributed by Petr Viktorin in :gh:`127987` and :cve:`2024-12718`.)
      2346  + * :func:`~tarfile.TarFile.extract` and :func:`~tarfile.TarFile.extractall`
      2347  +   now (re-)apply the extraction filter when substituting a link (hard or
      2348  +   symbolic) with a copy of another archive member, and when fixing up
      2349  +   directory attributes.
      2350  +   The former raises a new exception, :exc:`~tarfile.LinkFallbackError`.
      2351  +   (Contributed by Petr Viktorin for :cve:`2025-4330` and :cve:`2024-12718`.)
      2352  + * :func:`~tarfile.TarFile.extract` and :func:`~tarfile.TarFile.extractall`
```

```
2353 +   no longer extract rejected members when
2354 +   :func:`~tarfile.TarFile.errorlevel` is zero.
2355 +   (Contributed by Matt Prodani and Petr Viktorin in :gh:`112887`
2356 +   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', 'islink', 'samefile', 'sameopenfile',
11    -            'samestat']
   11 +            'samestat', 'ALLOW_MISSING']
12 12
13 13
14 14   # Does a path exist?
@@ -165,3 +165,12 @@ def _check_arg_types(funcname, *args):
165 165                                f'os.PathLike object, not
    {s.__class__.__name__!r}') from None
166 166       if hasstr and hasbytes:
167 167           raise TypeError("Can't mix strings and bytes in path components") from
    None
   168 +
   169 + # A singleton with a true boolean value.
   170 + @object.__new__
   171 + class ALLOW_MISSING:
   172 +     """Special value for use in realpath()."""
   173 +     def __repr__(self):
   174 +         return 'os.path.ALLOW_MISSING'
   175 +     def __reduce__(self):
   176 +         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", "isjunction"]
```

```
                +            "samefile", "sameopenfile", "samestat", "commonpath", "isjunction",
                +            "ALLOW_MISSING"]
34    35
35    36    def _get_bothseps(path):
36    37        if isinstance(path, bytes):
          @@ -609,9 +610,10 @@ def abspath(path):
609   610        from nt import _getfinalpathname, readlink as _nt_readlink
610   611    except ImportError:
611   612        # realpath is a no-op on systems without _getfinalpathname support.
612         -     realpath = abspath
              +     def realpath(path, *, strict=False):
              +         return abspath(path)
613   615    else:
614         -     def _readlink_deep(path):
              +     def _readlink_deep(path, ignored_error=OSError):
615   617            # These error codes indicate that we should stop reading links and
616   618            # return the path we currently have.
617   619            # 1: ERROR_INVALID_FUNCTION
          @@ -644,7 +646,7 @@ def _readlink_deep(path):
644   646                            path = old_path
645   647                            break
646   648                        path = normpath(join(dirname(old_path), path))
647         -             except OSError as ex:
              +             except ignored_error as ex:
648   650                     if ex.winerror in allowed_winerror:
649   651                         break
650   652                     raise
          @@ -653,7 +655,7 @@ def _readlink_deep(path):
653   655                     break
654   656            return path
655   657
656         - def _getfinalpathname_nonstrict(path):
              + def _getfinalpathname_nonstrict(path, ignored_error=OSError):
657   659        # These error codes indicate that we should stop resolving the path
658   660        # and return the value we currently have.
659   661        # 1: ERROR_INVALID_FUNCTION
          @@ -680,17 +682,18 @@ def _getfinalpathname_nonstrict(path):
680   682                try:
```

```
                    path = _getfinalpathname(path)
                    return join(path, tail) if tail else path
-               except OSError as ex:
+               except ignored_error as ex:
                    if ex.winerror not in allowed_winerror:
                        raise
                    try:
                        # The OS could not resolve this path fully, so we attempt
                        # to follow the link ourselves. If we succeed, join the tail
                        # and return.
-                       new_path = _readlink_deep(path)
+                       new_path = _readlink_deep(path,
+                                                 ignored_error=ignored_error)
                        if new_path != path:
                            return join(new_path, tail) if tail else new_path
-                   except OSError:
+                   except ignored_error:
                        # If we fail to readlink(), let's keep traversing
                        pass
                    path, name = split(path)
@@ -721,24 +724,32 @@ def realpath(path, *, strict=False):
                if normcase(path) == normcase(devnull):
                    return '\\\\.\\NUL'
            had_prefix = path.startswith(prefix)
+
+        if strict is ALLOW_MISSING:
+            ignored_error = FileNotFoundError
+            strict = True
+        elif strict:
+            ignored_error = ()
+        else:
+            ignored_error = OSError
+
            if not had_prefix and not isabs(path):
                path = join(cwd, path)
            try:
                path = _getfinalpathname(path)
                initial_winerror = 0
```

```
729  741            except ValueError as ex:
730  742                # gh-106242: Raised for embedded null characters
731       -            # In strict mode, we convert into an OSError.
     743  +            # In strict modes, we convert into an OSError.
732  744                # Non-strict mode returns the path as-is, since we've already
733  745                # made it absolute.
734  746                if strict:
735  747                    raise OSError(str(ex)) from None
736  748                path = normpath(path)
737       -        except OSError as ex:
738       -            if strict:
739       -                raise
     749  +        except ignored_error as ex:
740  750                initial_winerror = ex.winerror
741       -            path = _getfinalpathname_nonstrict(path)
     751  +            path = _getfinalpathname_nonstrict(path,
     752  +                                               ignored_error=ignored_error)
742  753            # The path returned by _getfinalpathname will always start with \\?\ -
743  754            # strip off that prefix unless it was already provided on the original
744  755            # 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", "isjunction"]
     38  +            "commonpath", "isjunction", "ALLOW_MISSING"]
39  39
40  40
41  41    def _get_sep(path):
@@ -438,6 +438,15 @@ def _joinrealpath(path, rest, strict, seen):
438 438            sep = '/'
439 439            curdir = '.'
440 440            pardir = '..'
     441  +         getcwd = os.getcwd
     442  +     if strict is ALLOW_MISSING:
     443  +         ignored_error = FileNotFoundError
```

```
        444 +     elif strict:
        445 +         ignored_error = ()
        446 +     else:
        447 +         ignored_error = OSError
        448 +
        449 +     maxlinks = None
441     450
442     451         if isabs(rest):
443     452             rest = rest[1:]
        @@ -460,9 +469,7 @@ def _joinrealpath(path, rest, strict, seen):
460     469             newpath = join(path, name)
461     470             try:
462     471                 st = os.lstat(newpath)
463         -         except OSError:
464         -             if strict:
465         -                 raise
        472 +         except ignored_error:
466     473                 is_link = False
467     474             else:
468     475                 is_link = stat.S_ISLNK(st.st_mode)
```

Comments 0