python / **cpython** Public

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

Commit `4633f3f`

6 people authored on Jun 3, 2025 · 57 / 58 · Partially verified

```
[3.11] gh-135034: Normalize link targets in tarfile, add
os.path.realpath(strict='allow_missing') (GH-135037) (GH-135068)

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

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>
```

3.11 (#98846, #135068) · v3.11.15 ••• v3.11.13

1 parent 2c6ca1a commit 4633f3f

11 files changed +1017 -138 lines changed

Top

Filter files…

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

11 files changed +1017 -138 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 *strict* is
356     -   ``True``, :exc:`OSError` is raised. If *strict* 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 :exc:`OSError`.
    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 *strict* is ``True``, the first error encountered when evaluating the
        path is
    367 +   re-raised.
    368 +   In particular, :exc:`FileNotFoundError` is raised if *path* does not exist,
    369 +   or another :exc:`OSError` if it is otherwise inaccessible.
    370 +
```

```
                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

```
                   @@ -239,6 +239,15 @@ The :mod:`tarfile` module defines the following
                   exceptions:
239             239        Raised to refuse extracting a symbolic link pointing outside the
                   destination
240             240        directory.
241             241
                242 + .. exception:: LinkFallbackError
                243 +
                244 +    Raised to refuse emulating a link (hard or symbolic) by extracting another
                245 +    archive member, when that member would be rejected by the filter location.
                246 +    The exception that was raised to reject the replacement member is
                   available
                247 +    as :attr:`!BaseException.__context__`.
                248 +
```

```
                249 +    .. versionadded:: next
                250 +
242             251
243             252     The following constants are available at the module level:
244             253
                    @@ -1037,6 +1046,12 @@ reused in custom filters:
1037            1046      Implements the ``'data'`` filter.
1038            1047      In addition to what ``tar_filter`` does:
1039            1048
                1049 +    - Normalize link targets (:attr:`TarInfo.linkname`) using
                1050 +      :func:`os.path.normpath`.
                1051 +      Note that this removes internal ``..`` components, which may change the
                1052 +      meaning of the link if the path in :attr:`!TarInfo.linkname` traverses
                1053 +      symbolic links.
                1054 +
1040            1055      - :ref:`Refuse <tarfile-extraction-refuse>` to extract links (hard or soft)
1041            1056        that link to absolute paths, or ones that link outside the destination.
1042            1057
                    @@ -1065,6 +1080,10 @@ reused in custom filters:
1065            1080
1066            1081      Return the modified ``TarInfo`` member.
1067            1082
                1083 +    .. versionchanged:: next
                1084 +
                1085 +       Link targets are now normalized.
                1086 +
1068            1087
1069            1088    .. _tarfile-extraction-refuse:
1070            1089
                    @@ -1091,6 +1110,7 @@ Here is an incomplete list of things to consider:
1091            1110    * Extract to a :func:`new temporary directory <tempfile.mkdtemp>`
1092            1111      to prevent e.g. exploiting pre-existing links, and to make it easier to
1093            1112      clean up after a failed extraction.
                1113 + * Disallow symbolic links if you do not need the functionality.
1094            1114    * When working with untrusted data, use external (e.g. OS-level) limits on
1095            1115      disk, memory and CPU usage.
1096            1116    * Check filenames against an allow-list of characters
```

Doc/whatsnew/3.11.rst

```
@@ -2786,3 +2786,37 @@ email
2786 2786   check if the *strict* paramater is available.
2787 2787   (Contributed by Thomas Dwyer and Victor Stinner for :gh:`102988` to improve
2788 2788   the CVE-2023-27043 fix.)
     2789 +
     2790 +
     2791 + Notable changes in 3.11.13
     2792 + ==========================
     2793 +
     2794 + os.path
     2795 + -------
     2796 +
     2797 + * The *strict* parameter to :func:`os.path.realpath` accepts a new value,
     2798 +   :data:`os.path.ALLOW_MISSING`.
     2799 +   If used, errors other than :exc:`FileNotFoundError` will be re-raised;
     2800 +   the resulting path can be missing but it will be free of symlinks.
     2801 +   (Contributed by Petr Viktorin for CVE 2025-4517.)
     2802 +
     2803 + tarfile
     2804 + -------
     2805 +
     2806 + * :func:`~tarfile.data_filter` now normalizes symbolic link targets in order
     to
     2807 +   avoid path traversal attacks.
     2808 +   (Contributed by Petr Viktorin in :gh:`127987` and CVE 2025-4138.)
     2809 + * :func:`~tarfile.TarFile.extractall` now skips fixing up directory
     attributes
     2810 +   when a directory was removed or replaced by another kind of file.
     2811 +   (Contributed by Petr Viktorin in :gh:`127987` and CVE 2024-12718.)
     2812 + * :func:`~tarfile.TarFile.extract` and :func:`~tarfile.TarFile.extractall`
     2813 +   now (re-)apply the extraction filter when substituting a link (hard or
     2814 +   symbolic) with a copy of another archive member, and when fixing up
     2815 +   directory attributes.
     2816 +   The former raises a new exception, :exc:`~tarfile.LinkFallbackError`.
     2817 +   (Contributed by Petr Viktorin for CVE 2025-4330 and CVE 2024-12718.)
     2818 + * :func:`~tarfile.TarFile.extract` and :func:`~tarfile.TarFile.extractall`
     2819 +   no longer extract rejected members when
     2820 +   :func:`~tarfile.TarFile.errorlevel` is zero.
```

```
2821 +   (Contributed by Matt Prodani and Petr Viktorin in :gh:`112887`
2822 +   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):
        @@ -578,9 +579,10 @@ def abspath(path):
578 579          from nt import _getfinalpathname, readlink as _nt_readlink
579 580      except ImportError:
580 581          # realpath is a no-op on systems without _getfinalpathname support.
581     -        realpath = abspath
    582 +        def realpath(path, *, strict=False):
    583 +            return abspath(path)
582 584      else:
583     -        def _readlink_deep(path):
    585 +        def _readlink_deep(path, ignored_error=OSError):
584 586              # These error codes indicate that we should stop reading links and
585 587              # return the path we currently have.
586 588              # 1: ERROR_INVALID_FUNCTION
        @@ -613,7 +615,7 @@ def _readlink_deep(path):
613 615                              path = old_path
614 616                              break
615 617                          path = normpath(join(dirname(old_path), path))
616     -                except OSError as ex:
    618 +                except ignored_error as ex:
617 619                      if ex.winerror in allowed_winerror:
618 620                          break
619 621                      raise
        @@ -622,7 +624,7 @@ def _readlink_deep(path):
622 624                      break
623 625              return path
624 626
625     -        def _getfinalpathname_nonstrict(path):
    627 +        def _getfinalpathname_nonstrict(path, ignored_error=OSError):
626 628              # These error codes indicate that we should stop resolving the path
627 629              # and return the value we currently have.
628 630              # 1: ERROR_INVALID_FUNCTION
        @@ -649,17 +651,18 @@ def _getfinalpathname_nonstrict(path):
649 651                  try:
650 652                      path = _getfinalpathname(path)
651 653                      return join(path, tail) if tail else path
```

```
652      -              except OSError as ex:
         654 +          except ignored_error as ex:
653      655                if ex.winerror not in allowed_winerror:
654      656                    raise
655      657                try:
656      658                    # The OS could not resolve this path fully, so we attempt
657      659                    # to follow the link ourselves. If we succeed, join the
                                tail
658      660                    # and return.
659      -                      new_path = _readlink_deep(path)
         661 +                  new_path = _readlink_deep(path,
         662 +                                            ignored_error=ignored_error)
660      663                    if new_path != path:
661      664                        return join(new_path, tail) if tail else new_path
662      -              except OSError:
         665 +          except ignored_error:
663      666                    # If we fail to readlink(), let's keep traversing
664      667                    pass
665      668                path, name = split(path)
                @@ -690,24 +693,32 @@ def realpath(path, *, strict=False):
690      693            if normcase(path) == normcase(devnull):
691      694                return '\\\\.\\NUL'
692      695        had_prefix = path.startswith(prefix)
         696 +
         697 +      if strict is ALLOW_MISSING:
         698 +          ignored_error = FileNotFoundError
         699 +          strict = True
         700 +      elif strict:
         701 +          ignored_error = ()
         702 +      else:
         703 +          ignored_error = OSError
         704 +
693      705        if not had_prefix and not isabs(path):
694      706            path = join(cwd, path)
695      707        try:
696      708            path = _getfinalpathname(path)
697      709            initial_winerror = 0
698      710        except ValueError as ex:
699      711            # gh-106242: Raised for embedded null characters
```

```
700       -             # In strict mode, we convert into an OSError.
          712 +         # In strict modes, we convert into an OSError.
701       713           # Non-strict mode returns the path as-is, since we've already
702       714           # made it absolute.
703       715           if strict:
704       716               raise OSError(str(ex)) from None
705       717           path = normpath(path)
706       -         except OSError as ex:
707       -             if strict:
708       -                 raise
          718 +         except ignored_error as ex:
709       719               initial_winerror = ex.winerror
710       -             path = _getfinalpathname_nonstrict(path)
          720 +             path = _getfinalpathname_nonstrict(path,
          721 +                                                ignored_error=ignored_error)
711       722           # The path returned by _getfinalpathname will always start with \\?\ -
712       723           # strip off that prefix unless it was already provided on the original
713       724           # 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):
          @@ -427,6 +427,15 @@ def _joinrealpath(path, rest, strict, seen):
427       427           sep = '/'
428       428           curdir = '.'
429       429           pardir = '..'
          430 +         getcwd = os.getcwd
          431 +     if strict is ALLOW_MISSING:
          432 +         ignored_error = FileNotFoundError
          433 +     elif strict:
          434 +         ignored_error = ()
```

```
                435 +     else:
                436 +         ignored_error = OSError
                437 +
                438 +     maxlinks = None
430             439
431             440       if isabs(rest):
432             441           rest = rest[1:]
                    @@ -449,9 +458,7 @@ def _joinrealpath(path, rest, strict, seen):
449             458           newpath = join(path, name)
450             459           try:
451             460               st = os.lstat(newpath)
452             -         except OSError:
453             -             if strict:
454             -                 raise
                461 +         except ignored_error:
455             462               is_link = False
456             463           else:
457             464               is_link = stat.S_ISLNK(st.st_mode)
```

Comments 0