


# vLLM using built-in hash() from Python 3.12 leads to predictable hash collisions in vLLM prefix cache

Low

 russellb published GHSA-rm76-4mrf-v9r8 2 days ago

| Package   | Affected versions | Patched versions |
|---|-------------------|------------------|
|  vllm (pip) | all               | >=0.7.2          |

### Severity

Low

 2.6 / 10

#### CVSS v3 base metrics

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

[Learn more about base metrics](#)

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

#### CVE ID

CVE-2025-25183

#### Weaknesses

No CWEs

#### Credits

 kexinoh

Reporter

### Description

#### Summary

Maliciously constructed prompts can lead to hash collisions, resulting in prefix cache reuse, which can interfere with subsequent responses and cause unintended behavior.

#### Details

vLLM's prefix caching makes use of Python's built-in hash() function. As of Python 3.12, the behavior of hash(None) has changed to be a predictable constant value. This makes it more feasible that someone could try exploit hash collisions.

#### Impact

The impact of a collision would be using cache that was generated using different content. Given knowledge of prompts in use and predictable hashing behavior, someone could intentionally populate the cache using a prompt known to collide with another prompt in use.

#### Solution

We address this problem by initializing hashes in vllm with a value that is no longer constant and predictable. It will be different each time vllm runs. This restores behavior we got in Python versions prior to 3.12.

Using a hashing algorithm that is less prone to collision (like sha256, for example) would be the best way to avoid the possibility of a collision. However, it would have an impact to both performance and memory footprint. Hash collisions may still occur, though they are no longer straight forward to predict.

To give an idea of the likelihood of a collision, for randomly generated hash values (assuming the hash generation built into Python is uniformly distributed), with a cache capacity of 50,000 messages and an average prompt length of 300, a collision will occur on average once every 1 trillion requests.

## References

- [#12621](#)
- [python/cpython@ 432117c](#)
- [python/cpython#99541](#)