

anthropics / anthropic-sdk-python Public

<> Code Issues 84 Pull requests 101 Discussions Actions Security and

Commit 715030c



dtmeadows authored and stainless-app[bot] committed 4 hours ago



fix(memory): use restrictive file mode for memory files

* fix(memory): use restrictive file mode for memory files

Extract `_FILE_CREATE_MODE` constant set to `0o600` (owner read/write only), replacing the previous `0o666`. In environments with a permissive umask (e.g. Docker where umask is often `0o000`), `0o666` would make memory files world-readable or even world-writable.

* fix(client): update mkdir calls to use `0o700`

main (#1264) · v0.87.0

1 parent [6cdbc5f](#) commit 715030c

1 file changed +21 -12 lines changed

↑ Top ⚙️

Filter files...

src/anthropic/lib/tools

_beta_builtin_memory_tool.py

1 file changed +21 -12 lines changed

Search within code ⚙️

```

...opic/lib/tools/_beta_builtin_memory_tool.py
@@ -42,6 +42,15 @@
42 42  MAX_LINES = 999999
43 43  LINE_NUMBER_WIDTH = len(str(MAX_LINES))
44 44
45 + # Owner read/write only. Avoids 0o666 which, in environments with a permissive
46 + # umask (e.g. Docker where umask is often 0o000), would make memory files
47 + # world-readable or even world-writable.
48 + _FILE_CREATE_MODE = 0o600

```

```

49 + # The default mkdir mode is 0o777, but we want to be more restrictive for
    memory
50 + # directories to avoid them being world-accessible in environments with
    permissive umasks
51 + # (eg Docker)
52 + _DIR_CREATE_MODE = 0o700
53 +
45 54
46 55     class BetaAbstractMemoryTool(BetaBuiltinFunctionTool):
47 56         """Abstract base class for memory tool implementations.
@@ -275,7 +284,7 @@ def _atomic_write_file(target_path: Path, content: str)
-> None:
275 284         data = content.encode("utf-8")
276 285
277 286         try:
278 -             fd = os.open(temp_path, os.O_CREAT | os.O_EXCL | os.O_WRONLY, 0o666)
287 +             fd = os.open(temp_path, os.O_CREAT | os.O_EXCL | os.O_WRONLY,
                _FILE_CREATE_MODE)
279 288         try:
280 289             offset = 0
281 290             while offset < len(data):
@@ -342,7 +351,7 @@ def __init__(self, base_path: str = "./memory"):
342 351         super().__init__()
343 352         self.base_path = Path(base_path)
344 353         self.memory_root = self.base_path / "memories"
345 -         self.memory_root.mkdir(parents=True, exist_ok=True)
354 +         self.memory_root.mkdir(parents=True, exist_ok=True,
                mode=_DIR_CREATE_MODE)
346 355
347 356         def _validate_path(self, path: str) -> Path:
348 357             """Validate and resolve memory paths"""
@@ -434,10 +443,10 @@ def collect_items(dir_path: Path, relative_path: str,
depth: int) -> None:
434 443         def create(self, command: BetaMemoryTool20250818CreateCommand) -> str:
435 444             full_path = self._validate_path(command.path)
436 445
437 -             full_path.parent.mkdir(parents=True, exist_ok=True)
446 +             full_path.parent.mkdir(parents=True, exist_ok=True,
                mode=_DIR_CREATE_MODE)

```

```


438 447
439 448         try:
440 -         fd = os.open(full_path, os.O_CREAT | os.O_EXCL | os.O_WRONLY,
          0o666)
449 +         fd = os.open(full_path, os.O_CREAT | os.O_EXCL | os.O_WRONLY,
          _FILE_CREATE_MODE)
441 450         try:
442 451             os.write(fd, command.file_text.encode("utf-8"))
443 452             os.fsync(fd)
444 453
445 454     @@ -549,7 +558,7 @@ def rename(self, command:
446 455     BetaMemoryTool20250818RenameCommand) -> str:
447 456
448 457         if new_full_path.exists():
449 458             raise ToolError(f"The destination {command.new_path} already
450 459             exists")
451 460
452 461     -         new_full_path.parent.mkdir(parents=True, exist_ok=True)
453 462     +         new_full_path.parent.mkdir(parents=True, exist_ok=True,
454 463             mode=_DIR_CREATE_MODE)
455 464
456 465         try:
457 466             old_full_path.rename(new_full_path)
458 467
459 468     @@ -563,7 +572,7 @@ def clear_all_memory(self) -> str:
460 469     """Override the base implementation to provide file system clearing."""
461 470     if self.memory_root.exists():
462 471         shutil.rmtree(self.memory_root)
463 472     -         self.memory_root.mkdir(parents=True, exist_ok=True)
464 473     +         self.memory_root.mkdir(parents=True, exist_ok=True,
465 474             mode=_DIR_CREATE_MODE)
466 475
467 476         return "All memory cleared"
468 477
469 478
470 479     @@ -576,7 +585,7 @@ async def _async_atomic_write_file(target_path:
471 480     AsyncPath, content: str) -> None
472 481
473 482         try:
474 483
475 484             def write_replace_and_sync() -> None:
476 485     -         fd = os.open(sync_temp_path, os.O_CREAT | os.O_EXCL | os.O_WRONLY,
477 486             0o666)
478 487     +         fd = os.open(sync_temp_path, os.O_CREAT | os.O_EXCL | os.O_WRONLY,
479 488             _FILE_CREATE_MODE)

```

```

580 589         try:
581 590             offset = 0
582 591         while offset < len(data):
@@ -624,7 +633,7 @@ def __init__(self, base_path: str = "./memory"):
624 633
625 634         async def _ensure_memory_root(self) -> None:
626 635             """Ensure the memory root directory exists"""
627 -         await self.memory_root.mkdir(parents=True, exist_ok=True)
636 +         await self.memory_root.mkdir(parents=True, exist_ok=True,
mode=_DIR_CREATE_MODE)
628 637
629 638         async def _validate_path(self, path: str) -> AsyncPath:
630 639             """Validate and resolve memory paths"""
@@ -724,13 +733,13 @@ async def create(self, command:
BetaMemoryTool20250818CreateCommand) -> str:
724 733         await self._ensure_memory_root()
725 734         full_path = await self._validate_path(command.path)
726 735
727 -         await full_path.parent.mkdir(parents=True, exist_ok=True)
736 +         await full_path.parent.mkdir(parents=True, exist_ok=True,
mode=_DIR_CREATE_MODE)
728 737
729 738         try:
730 739             sync_full_path = Path(str(full_path))
731 740
732 741             def create_exclusive() -> None:
733 -                 fd = os.open(sync_full_path, os.O_CREAT | os.O_EXCL |
os.O_WRONLY, 0o666)
742 +                 fd = os.open(sync_full_path, os.O_CREAT | os.O_EXCL |
os.O_WRONLY, _FILE_CREATE_MODE)
734 743             try:
735 744                 os.write(fd, command.file_text.encode("utf-8"))
736 745                 os.fsync(fd)
@@ -848,7 +857,7 @@ async def rename(self, command:
BetaMemoryTool20250818RenameCommand) -> str:
848 857         if await new_full_path.exists():
849 858             raise ToolError(f"The destination {command.new_path} already
exists")
850 859

```

851	-	<code>await new_full_path.parent.mkdir(parents=True, exist_ok=True)</code>
860	+	<code>await new_full_path.parent.mkdir(parents=True, exist_ok=True, mode=_DIR_CREATE_MODE)</code>
852	861	
853	862	<code>try:</code>
854	863	<code>await old_full_path.rename(new_full_path)</code>
		<code>@@ -862,5 +871,5 @@ async def clear_all_memory(self) -> str:</code>
862	871	<code>"""Override the base implementation to provide file system clearing."""</code>
863	872	<code>if await self.memory_root.exists():</code>
864	873	<code>await run_sync(shutil.rmtree, str(self.memory_root))</code>
865	-	<code>await self.memory_root.mkdir(parents=True, exist_ok=True)</code>
874	+	<code>await self.memory_root.mkdir(parents=True, exist_ok=True, mode=_DIR_CREATE_MODE)</code>
866	875	<code>return "All memory cleared"</code>

Comments 0



Please [sign in](#) to comment.