anthropics / **anthropic-sdk-python** Public

Code | Issues 88 | Pull requests 104 | Discussions | Actions | Security an

Commit 715030c

dtmeadows authored and **stainless-app[bot]** committed 4 days 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) · v0.89.0 ••• 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)
          @@ -549,7 +558,7 @@ def rename(self, command:
          BetaMemoryTool20250818RenameCommand) -> str:
549   558            if new_full_path.exists():
550   559                raise ToolError(f"The destination {command.new_path} already
          exists")
551   560
552       -          new_full_path.parent.mkdir(parents=True, exist_ok=True)
      561 +          new_full_path.parent.mkdir(parents=True, exist_ok=True,
          mode=_DIR_CREATE_MODE)
553   562
554   563            try:
555   564                old_full_path.rename(new_full_path)
          @@ -563,7 +572,7 @@ def clear_all_memory(self) -> str:
563   572            """Override the base implementation to provide file system clearing."""
564   573            if self.memory_root.exists():
565   574                shutil.rmtree(self.memory_root)
566       -          self.memory_root.mkdir(parents=True, exist_ok=True)
      575 +          self.memory_root.mkdir(parents=True, exist_ok=True,
          mode=_DIR_CREATE_MODE)
567   576            return "All memory cleared"
568   577
569   578
          @@ -576,7 +585,7 @@ async def _async_atomic_write_file(target_path:
          AsyncPath, content: str) -> None
576   585        try:
577   586
578   587            def write_replace_and_sync() -> None:
579       -              fd = os.open(sync_temp_path, os.O_CREAT | os.O_EXCL | os.O_WRONLY,
          0o666)
```

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

Comments 0

Please sign in to comment.