volcengine / **OpenViking** Public

Code | Issues 80 | Pull requests 40 | Discussions | Actions | Projects

Commit 46b3e76

13ernkastel authored on Feb 28 · 15 / 15 · Verified

```
Refine ovpack import validation and client security tests (#344)
```

main (#344) · v0.3.3 ••• v0.2.1

1 parent 04dff46 commit 46b3e76

2 files changed +95 -4 lines changed

Top

Filter files…

- openviking/storage
  - local_fs.py
- tests/client
  - test_import_export.py

2 files changed +95 -4 lines changed

Search within code

openviking/storage/local_fs.py

```
@@ -2,6 +2,7 @@
2  2  # SPDX-License-Identifier: Apache-2.0
3  3  import json
4  4  import os
   5  + import re
5  6  import zipfile
6  7  from datetime import datetime
7  8  from typing import cast
@@ -16,6 +17,32 @@
16 17  logger = get_logger(__name__)
17 18
```

```
18  19
        20  + _UNSAFE_PATH_RE = re.compile(r"(^|[\\/])\.\.($|[\\/])")
        21  + _DRIVE_RE = re.compile(r"^[A-Za-z]:")
        22  +
        23  +
        24  + def _validate_ovpack_member_path(zip_path: str, base_name: str) -> str:
        25  +     """Validate a zip member path for ovpack imports and reject unsafe
            entries."""
        26  +     if not zip_path:
        27  +         raise ValueError("Invalid ovpack entry: empty path")
        28  +     if "\\" in zip_path:
        29  +         raise ValueError(f"Unsafe ovpack entry path: {zip_path!r}")
        30  +     if zip_path.startswith("/"):
        31  +         raise ValueError(f"Unsafe ovpack entry path: {zip_path!r}")
        32  +     if _DRIVE_RE.match(zip_path):
        33  +         raise ValueError(f"Unsafe ovpack entry path: {zip_path!r}")
        34  +     if _UNSAFE_PATH_RE.search(zip_path):
        35  +         raise ValueError(f"Unsafe ovpack entry path: {zip_path!r}")
        36  +
        37  +     parts = zip_path.split("/")
        38  +     if any(part == ".." for part in parts):
        39  +         raise ValueError(f"Unsafe ovpack entry path: {zip_path!r}")
        40  +     if not parts or parts[0] != base_name:
        41  +         raise ValueError(f"Invalid ovpack entry root: {zip_path!r}")
        42  +
        43  +     return zip_path
        44  +
        45  +
19  46      def ensure_ovpack_extension(path: str) -> str:
20  47          """Ensure path ends with .ovpack extension."""
21  48          if not path.endswith(".ovpack"):
            @@ -174,19 +201,21 @@ async def import_ovpack(
174 201                  if not zip_path:
175 202                      continue
176 203
        204 +                safe_zip_path = _validate_ovpack_member_path(zip_path, base_name)
        205 +
177 206                  # Handle directory entries
178     -                if zip_path.endswith("/"):
```

```
179     -                   rel_path = get_viking_rel_path_from_zip(zip_path.rstrip("/"))
        207 +           if safe_zip_path.endswith("/"):
        208 +               rel_path =
            get_viking_rel_path_from_zip(safe_zip_path.rstrip("/"))
180     209                 target_dir_uri = f"{root_uri}/{rel_path}" if rel_path else
            root_uri
181     210                 await viking_fs.mkdir(target_dir_uri, exist_ok=True, ctx=ctx)
182     211                 continue
183     212
184     213             # Handle file entries
185     -               rel_path = get_viking_rel_path_from_zip(zip_path)
        214 +           rel_path = get_viking_rel_path_from_zip(safe_zip_path)
186     215             target_file_uri = f"{root_uri}/{rel_path}" if rel_path else
            root_uri
187     216
188     217             try:
189     -                   data = zf.read(zip_path)
        218 +               data = zf.read(safe_zip_path)
190     219                 await viking_fs.write_file_bytes(target_file_uri, data,
            ctx=ctx)
191     220             except Exception as e:
192     221                 logger.error(f"Failed to import {zip_path} to
            {target_file_uri}: {e}")
```

tests/client/test_import_export.py

```
            @@ -3,8 +3,12 @@
3   3       """Import/export tests"""
4   4
5   5
        6   + import io
        7   + import zipfile
6   8       from pathlib import Path
7   9
        10  + import pytest
        11  +
8   12      from openviking import AsyncOpenViking
9   13
10  14
```

```
@@ -111,3 +115,61 @@ async def test_import_export_roundtrip(
111 115
112 116            # Verify content consistency
113 117            assert original_content == imported_content
    118 +
    119 +
    120 +    @staticmethod
    121 +    def _build_ovpack(zip_path: Path, entries: dict[str, str]) -> None:
    122 +        buffer = io.BytesIO()
    123 +        with zipfile.ZipFile(buffer, "w") as zf:
    124 +            for name, content in entries.items():
    125 +                zf.writestr(name, content)
    126 +        zip_path.write_bytes(buffer.getvalue())
    127 +
    128 +    @pytest.mark.parametrize(
    129 +        "entries,error_pattern",
    130 +        [
    131 +            (
    132 +                {
    133 +                    "pkg/_._meta.json": '{"uri": "viking://resources/pkg"}',
    134 +                    "pkg/../../escape.txt": "pwned",
    135 +                },
    136 +                "Unsafe ovpack entry path",
    137 +            ),
    138 +            (
    139 +                {
    140 +                    "pkg/_._meta.json": '{"uri": "viking://resources/pkg"}',
    141 +                    "/abs/path.txt": "pwned",
    142 +                },
    143 +                "Unsafe ovpack entry path",
    144 +            ),
    145 +            (
    146 +                {
    147 +                    "pkg/_._meta.json": '{"uri": "viking://resources/pkg"}',
    148 +                    "C:/drive/path.txt": "pwned",
    149 +                },
    150 +                "Unsafe ovpack entry path",
    151 +            ),
    152 +            (
    153 +                {
```

```
154 +                     "pkg/_._meta.json": '{"uri": "viking://resources/pkg"}',
155 +                     "pkg\\windows\\path.txt": "pwned",
156 +                 },
157 +                 "Unsafe ovpack entry path",
158 +             ),
159 +             (
160 +                 {
161 +                     "pkg/_._meta.json": '{"uri": "viking://resources/pkg"}',
162 +                     "other/file.txt": "pwned",
163 +                 },
164 +                 "Invalid ovpack entry root",
165 +             ),
166 +         ],
167 +     )
168 +     async def test_import_rejects_unsafe_entries(
169 +         self, client: AsyncOpenViking, temp_dir: Path, entries: dict[str, str],
    error_pattern: str
170 +     ):
171 +         ovpack_path = temp_dir / "malicious.ovpack"
172 +         self._build_ovpack(ovpack_path, entries)
173 +
174 +         with pytest.raises(ValueError, match=error_pattern):
175 +             await client.import_ovpack(str(ovpack_path),
    "viking://resources/security/", vectorize=False)
```

Comments 0

Please sign in to comment.