HKUDS / **OpenHarness** Public

Code | Issues 7 | Pull requests 12 | Actions | Projects | Security and quali

Commit dd1d235

Hinotoi-agent authored 3 days ago · ✕ 2 / 4 · Verified

```
fix: harden gateway slash command security (#127)

* fix: harden gateway slash command security

* fix: add secure remote admin opt-in for gateway commands
```

main (#127)

1 parent 63ce793 commit dd1d235

8 files changed +285 -7 lines changed

Top

Filter files…

- ohmo
  - cli.py
  - gateway
    - models.py
    - runtime.py
    - service.py
  - workspace.py
- src/openharness/commands
  - registry.py
- tests
  - test_commands
    - test_registry.py
  - test_ohmo
    - test_gateway.py

8 files changed +285 -7 lines changed

Search within code

ohmo/cli.py

```
@@ -306,13 +306,31 @@ def _run_gateway_config_wizard(workspace: str | Path)
-> GatewayConfig:
306 306            "Send tool hints to channels?",
307 307            default=existing.send_tool_hints,
308 308        )
    309 +      allow_remote_admin_commands = _confirm_prompt(
    310 +          "Allow explicitly listed administrative slash commands from remote
channels?",
    311 +          default=existing.allow_remote_admin_commands,
    312 +      )
    313 +      default_allowlist = ", ".join(existing.allowed_remote_admin_commands)
    314 +      allowed_remote_admin_commands: list[str] = []
    315 +      if allow_remote_admin_commands:
    316 +          allowlist_raw = _text_prompt(
    317 +              "Allowed remote admin commands (comma-separated, e.g. permissions,
plan)",
    318 +              default=default_allowlist,
    319 +          )
    320 +          allowed_remote_admin_commands = [
    321 +              item.strip().lstrip("/")
    322 +              for item in allowlist_raw.split(",")
    323 +              if item.strip()
    324 +          ]
309 325        config = existing.model_copy(
310 326            update={
311 327                "provider_profile": provider_profile,
312 328                "enabled_channels": enabled_channels,
313 329                "channel_configs": channel_configs,
314 330                "send_progress": send_progress,
315 331                "send_tool_hints": send_tool_hints,
    332 +              "allow_remote_admin_commands": allow_remote_admin_commands,
    333 +              "allowed_remote_admin_commands": allowed_remote_admin_commands,
316 334            }
317 335        )
318 336        save_gateway_config(config, workspace)
```

```
@@ -328,6 +346,13 @@ def _print_gateway_config_summary(config: GatewayConfig) -> None:
328 346             )
329 347         else:
330 348             print(f"Configured provider_profile={config.provider_profile}; no channels enabled yet.")
    349 +     if config.allow_remote_admin_commands and config.allowed_remote_admin_commands:
    350 +         print(
    351 +             "Remote admin opt-in enabled for: "
    352 +             + ", ".join(f"/{name}" for name in config.allowed_remote_admin_commands)
    353 +         )
    354 +     else:
    355 +         print("Remote admin commands remain local-only.")
331 356
332 357
333 358 def _maybe_restart_gateway(*, cwd: str | Path, workspace: str | Path) -> None:
```

ohmo/gateway/models.py

```
@@ -15,6 +15,8 @@ class GatewayConfig(BaseModel):
15 15     send_tool_hints: bool = True
16 16     permission_mode: str = "default"
17 17     sandbox_enabled: bool = False
   18 +     allow_remote_admin_commands: bool = False
   19 +     allowed_remote_admin_commands: list[str] = Field(default_factory=list)
18 20     log_level: str = "INFO"
19 21     channel_configs: dict[str, dict] = Field(default_factory=dict)
20 22
```

ohmo/gateway/runtime.py

```
@@ -12,7 +12,7 @@
12 12 import string
13 13
14 14 from openharness.channels.bus.events import InboundMessage
15    - from openharness.commands import CommandContext
   15 + from openharness.commands import CommandContext, CommandResult
```

```
16  16    from openharness.engine.messages import ConversationMessage, ImageBlock,
          TextBlock
17  17    from openharness.engine.query import MaxTurnsExceeded
18  18    from openharness.engine.stream_events import (
          @@ -27,6 +27,7 @@
27  27    from openharness.prompts import build_runtime_system_prompt
28  28    from openharness.ui.runtime import RuntimeBundle, _last_user_text,
          build_runtime, close_runtime, start_runtime
29  29
      30  + from ohmo.gateway.config import load_gateway_config
30  31    from ohmo.prompts import build_ohmo_system_prompt
31  32    from ohmo.session_storage import OhmoSessionBackend
32  33    from ohmo.workspace import get_plugins_dir, get_skills_dir,
          initialize_workspace
          @@ -81,13 +82,26 @@ def __init__(
81  82            self._model = model
82  83            self._max_turns = max_turns
83  84            self._workspace = initialize_workspace(workspace)
      85  +         self._gateway_config = load_gateway_config(self._workspace)
84  86            self._session_backend = OhmoSessionBackend(self._workspace)
85  87            self._bundles: dict[str, RuntimeBundle] = {}
86  88
87  89        @property
88  90        def active_sessions(self) -> int:
89  91            return len(self._bundles)
90  92
      93  +     def _remote_admin_allowed(self, command) -> bool:
      94  +         if not getattr(command, "remote_admin_opt_in", False):
      95  +             return False
      96  +         if not self._gateway_config.allow_remote_admin_commands:
      97  +             return False
      98  +         allowed = {
      99  +             str(name).strip().lower()
     100  +             for name in self._gateway_config.allowed_remote_admin_commands
     101  +             if str(name).strip()
     102  +         }
     103  +         return command.name.lower() in allowed
     104  +
```

```
91  105       async def get_bundle(self, session_key: str, latest_user_prompt: str | None
                = None) -> RuntimeBundle:
92  106           """Return an existing bundle or create a new one."""
93  107           bundle = self._bundles.get(session_key)
    @@ -149,6 +163,29 @@ async def stream_message(self, message: InboundMessage,
    session_key: str):
149 163           parsed = bundle.commands.lookup(user_prompt)
150 164           if parsed is not None and not message.media:
151 165               command, args = parsed
    166 +             remote_allowed = getattr(command, "remote_invocable", True)
    167 +             if not remote_allowed and self._remote_admin_allowed(command):
    168 +                 remote_allowed = True
    169 +                 logger.warning(
    170 +                     "ohmo gateway remote administrative command accepted
                    channel=%s chat_id=%s sender_id=%s command=%s",
    171 +                     message.channel,
    172 +                     message.chat_id,
    173 +                     message.sender_id,
    174 +                     command.name,
    175 +                 )
    176 +             if not remote_allowed:
    177 +                 result = CommandResult(
    178 +                     message=f"/{command.name} is only available in the local
                    OpenHarness UI."
    179 +                 )
    180 +                 async for update in self._stream_command_result(
    181 +                     bundle=bundle,
    182 +                     message=message,
    183 +                     session_key=session_key,
    184 +                     user_prompt=user_prompt,
    185 +                     result=result,
    186 +                 ):
    187 +                     yield update
    188 +                 return
152 189           result = await command.handler(
153 190               args,
154 191               CommandContext(
```

ohmo/gateway/service.py

```
@@ -43,6 +43,11 @@ def __init__(self, cwd: str | Path | None = None,
workspace: str | Path | None =
43  43          root = initialize_workspace(self._workspace)
44  44          os.environ["OHMO_WORKSPACE"] = str(root)
45  45          self._config = load_gateway_config(self._workspace)
    46  +       if self._config.allow_remote_admin_commands and
                self._config.allowed_remote_admin_commands:
    47  +           logger.warning(
    48  +               "ohmo gateway remote administrative commands enabled
                commands=%s",
    49  +               ",".join(self._config.allowed_remote_admin_commands),
    50  +           )
46  51          self._bus = MessageBus()
47  52          self._runtime_pool = OhmoSessionRuntimePool(
48  53              cwd=self._cwd,
```

ohmo/workspace.py

```
@@ -283,6 +283,8 @@ def initialize_workspace(workspace: str | Path | None =
None) -> Path:
283 283                     "send_tool_hints": True,
284 284                     "permission_mode": "default",
285 285                     "sandbox_enabled": False,
    286 +                   "allow_remote_admin_commands": False,
    287 +                   "allowed_remote_admin_commands": [],
286 288                     "log_level": "INFO",
287 289                     "channel_configs": {},
288 290                 },
```

src/openharness/commands/registry.py

```
@@ -100,6 +100,8 @@ class SlashCommand:
100 100     name: str
101 101     description: str
102 102     handler: CommandHandler
    103 +   remote_invocable: bool = True
    104 +   remote_admin_opt_in: bool = False
103 105
104 106
```

```
105  107      class CommandRegistry:
          @@ -375,9 +377,9 @@ async def _memory_handler(args: str, context:
          CommandContext) -> CommandResult:
375  377              return CommandResult(message="\n".join(path.name for path in
          memory_files))
376  378          if action == "show" and rest:
377  379              memory_dir = get_project_memory_dir(context.cwd)
378       -          path = memory_dir / rest
379       -          if not path.exists():
380       -              path = memory_dir / f"{rest}.md"
     380  +          path = _resolve_memory_entry_path(memory_dir, rest)
     381  +          if path is None:
     382  +              return CommandResult(message="Memory entry path must stay
          within the project memory directory.")
381  383              if not path.exists():
382  384                  return CommandResult(message=f"Memory entry not found:
          {rest}")
383  385              return CommandResult(message=path.read_text(encoding="utf-8"))
          @@ -1560,8 +1562,24 @@ async def _tasks_handler(args: str, context:
          CommandContext) -> CommandResult:
1560 1562      registry.register(SlashCommand("mcp", "Show MCP status", _mcp_handler))
1561 1563      registry.register(SlashCommand("plugin", "Manage plugins",
          _plugin_handler))
1562 1564      registry.register(SlashCommand("reload-plugins", "Reload plugin discovery
          for this workspace", _reload_plugins_handler))
1563      -    registry.register(SlashCommand("permissions", "Show or update permission
          mode", _permissions_handler))
1564      -    registry.register(SlashCommand("plan", "Toggle plan permission mode",
          _plan_handler))
     1565 +    registry.register(
     1566 +        SlashCommand(
     1567 +            "permissions",
     1568 +            "Show or update permission mode",
     1569 +            _permissions_handler,
     1570 +            remote_invocable=False,
     1571 +            remote_admin_opt_in=True,
     1572 +        )
     1573 +    )
     1574 +    registry.register(
     1575 +        SlashCommand(
```

```
                1576 +             "plan",
                1577 +             "Toggle plan permission mode",
                1578 +             _plan_handler,
                1579 +             remote_invocable=False,
                1580 +             remote_admin_opt_in=True,
                1581 +         )
                1582 +     )
1565            1583         registry.register(SlashCommand("fast", "Show or update fast mode",
                             _fast_handler))
1566            1584         registry.register(SlashCommand("effort", "Show or update reasoning
                             effort", _effort_handler))
1567            1585         registry.register(SlashCommand("passes", "Show or update reasoning pass
                             count", _passes_handler))
                             @@ -1618,3 +1636,28 @@ async def _plugin_command_handler(
1618            1636                 )
1619            1637             )
1620            1638         return registry
                1639 +
                1640 +
                1641 + def _resolve_memory_entry_path(memory_dir: Path, candidate: str) -> Path |
                             None:
                1642 +     """Resolve a memory entry path while enforcing containment under
                             ``memory_dir``."""
                1643 +
                1644 +     base = memory_dir.resolve()
                1645 +     resolved = _resolve_memory_candidate(base, candidate)
                1646 +     if resolved is not None and resolved.exists():
                1647 +         return resolved
                1648 +     fallback = _resolve_memory_candidate(base, f"{candidate}.md")
                1649 +     if fallback is not None:
                1650 +         return fallback
                1651 +     return None
                1652 +
                1653 +
                1654 + def _resolve_memory_candidate(memory_dir: Path, candidate: str) -> Path |
                             None:
                1655 +     path = Path(candidate).expanduser()
                1656 +     if not path.is_absolute():
                1657 +         path = memory_dir / path
```

```
1658 +     resolved = path.resolve()
1659 +     try:
1660 +         resolved.relative_to(memory_dir)
1661 +     except ValueError:
1662 +         return None
1663 +     return resolved
```

tests/test_commands/test_registry.py

```
@@ -75,6 +75,53 @@ async def test_permissions_command_persists(tmp_path: Path, monkeypatch):
75 75         assert load_settings().permission.mode == "full_auto"
76 76
77 77
78 + @pytest.mark.asyncio
79 + async def test_permissions_command_is_marked_local_only(tmp_path: Path, monkeypatch):
80 +     monkeypatch.setenv("OPENHARNESS_CONFIG_DIR", str(tmp_path / "config"))
81 +     registry = create_default_command_registry()
82 +     command, _ = registry.lookup("/permissions set full_auto")
83 +     assert command is not None
84 +     assert command.remote_invocable is False
85 +
86 +
87 + @pytest.mark.asyncio
88 + async def test_permissions_command_supports_explicit_remote_admin_opt_in(tmp_path: Path, monkeypatch):
89 +     monkeypatch.setenv("OPENHARNESS_CONFIG_DIR", str(tmp_path / "config"))
90 +     registry = create_default_command_registry()
91 +     command, _ = registry.lookup("/permissions set full_auto")
92 +     assert command is not None
93 +     assert getattr(command, "remote_admin_opt_in", False) is True
94 +
95 +
96 + @pytest.mark.asyncio
97 + async def test_memory_show_rejects_path_traversal(tmp_path: Path, monkeypatch):
98 +     monkeypatch.setenv("OPENHARNESS_CONFIG_DIR", str(tmp_path / "config"))
99 +     monkeypatch.setenv("OPENHARNESS_DATA_DIR", str(tmp_path / "data"))
100 +     registry = create_default_command_registry()
101 +     command, args = registry.lookup("/memory show ../../../../../../etc/hosts")
```

```
+     assert command is not None
+
+     result = await command.handler(args,
CommandContext(engine=_make_engine(tmp_path), cwd=str(tmp_path)))
+
+     assert result.message == "Memory entry path must stay within the project
memory directory."
+
+
+ @pytest.mark.asyncio
+ async def test_memory_show_reads_normal_entries_with_md_fallback(tmp_path:
Path, monkeypatch):
+     monkeypatch.setenv("OPENHARNESS_CONFIG_DIR", str(tmp_path / "config"))
+     monkeypatch.setenv("OPENHARNESS_DATA_DIR", str(tmp_path / "data"))
+     registry = create_default_command_registry()
+
+     add_command, add_args = registry.lookup("/memory add Notes :: hello world")
+     assert add_command is not None
+     await add_command.handler(add_args,
CommandContext(engine=_make_engine(tmp_path), cwd=str(tmp_path)))
+
+     show_command, show_args = registry.lookup("/memory show Notes")
+     result = await show_command.handler(show_args,
CommandContext(engine=_make_engine(tmp_path), cwd=str(tmp_path)))
+
+     assert "hello world" in result.message
+
+
 @pytest.mark.asyncio
 async def test_model_command_persists(tmp_path: Path, monkeypatch):
     monkeypatch.setenv("OPENHARNESS_CONFIG_DIR", str(tmp_path / "config"))
```

tests/test_ohmo/test_gateway.py

```
@@ -12,11 +12,13 @@
 from openharness.channels.bus.events import InboundMessage
 from openharness.channels.bus.queue import MessageBus
 from openharness.commands import CommandResult
+from openharness.commands.registry import SlashCommand
```

```
15  16    from openharness.engine.messages import ConversationMessage, ImageBlock,
          TextBlock
16  17    from openharness.engine.stream_events import AssistantTextDelta,
          CompactProgressEvent, ToolExecutionStarted
17  18
18  19    from ohmo.gateway.bridge import OhmoGatewayBridge, _format_gateway_error
19      - from ohmo.gateway.models import GatewayState
    20  + from ohmo.gateway.config import save_gateway_config
    21  + from ohmo.gateway.models import GatewayConfig, GatewayState
20  22    from ohmo.gateway.runtime import OhmoSessionRuntimePool,
          _build_inbound_user_message, _format_channel_progress
21  23    from ohmo.gateway.service import OhmoGatewayService, gateway_status,
          stop_gateway_process
22  24    from ohmo.gateway.router import session_key_for_message
        ↓ ↑ @@ -356,6 +358,121 @@ async def fake_start_runtime(bundle):
356 358         assert updates[1].text.startswith("🛠️ Using web_fetch")
357 359
358 360
    361 + @pytest.mark.asyncio
    362 + async def
          test_runtime_pool_blocks_local_only_commands_from_remote_messages(tmp_path,
          monkeypatch):
    363 +     workspace = tmp_path / ".ohmo-home"
    364 +     initialize_workspace(workspace)
    365 +     handler_called = False
    366 +
    367 +     async def forbidden_handler(args, context):
    368 +         nonlocal handler_called
    369 +         handler_called = True
    370 +         return CommandResult(message="should not run")
    371 +
    372 +     async def fake_build_runtime(**kwargs):
    373 +         class FakeEngine:
    374 +             messages = []
    375 +             total_usage = UsageSnapshot()
    376 +
    377 +             def set_system_prompt(self, prompt):
    378 +                 return None
    379 +
```

```
380 +         command = SlashCommand(
381 +             "permissions",
382 +             "Show or update permission mode",
383 +             forbidden_handler,
384 +             remote_invocable=False,
385 +         )
386 +         command.remote_admin_opt_in = True
387 +         return SimpleNamespace(
388 +             engine=FakeEngine(),
389 +             session_id="sess123",
390 +             current_settings=lambda: SimpleNamespace(model="gpt-5.4"),
391 +             commands=SimpleNamespace(lookup=lambda raw: (command,
    "full_auto")),
392 +         )
393 +
394 +     async def fake_start_runtime(bundle):
395 +         return None
396 +
397 +     monkeypatch.setattr("ohmo.gateway.runtime.build_runtime",
    fake_build_runtime)
398 +     monkeypatch.setattr("ohmo.gateway.runtime.start_runtime",
    fake_start_runtime)
399 +
400 +     pool = OhmoSessionRuntimePool(cwd=tmp_path, workspace=workspace,
    provider_profile="codex")
401 +     message = InboundMessage(channel="feishu", sender_id="u1", chat_id="c1",
    content="/permissions full_auto")
402 +     updates = [u async for u in pool.stream_message(message, "feishu:c1")]
403 +
404 +     assert handler_called is False
405 +     assert updates[-1].kind == "final"
406 +     assert updates[-1].text == "/permissions is only available in the local
    OpenHarness UI."
407 +
408 +
409 + @pytest.mark.asyncio
410 + async def test_runtime_pool_allows_opted_in_remote_admin_commands(tmp_path,
    monkeypatch, caplog):
411 +     workspace = tmp_path / ".ohmo-home"
412 +     initialize_workspace(workspace)
```

```
413 +     save_gateway_config(
414 +         GatewayConfig(
415 +             provider_profile="codex",
416 +             allow_remote_admin_commands=True,
417 +             allowed_remote_admin_commands=["permissions"],
418 +         ),
419 +         workspace,
420 +     )
421 +     handler_called = False
422 +
423 +     async def allowed_handler(args, context):
424 +         nonlocal handler_called
425 +         handler_called = True
426 +         return CommandResult(message=f"ran with {args}")
427 +
428 +     async def fake_build_runtime(**kwargs):
429 +         class FakeEngine:
430 +             messages = []
431 +             total_usage = UsageSnapshot()
432 +
433 +             def set_system_prompt(self, prompt):
434 +                 return None
435 +
436 +         command = SlashCommand(
437 +             "permissions",
438 +             "Show or update permission mode",
439 +             allowed_handler,
440 +             remote_invocable=False,
441 +         )
442 +         command.remote_admin_opt_in = True
443 +         return SimpleNamespace(
444 +             engine=FakeEngine(),
445 +             session_id="sess123",
446 +             current_settings=lambda: SimpleNamespace(model="gpt-5.4"),
447 +             commands=SimpleNamespace(lookup=lambda raw: (command,
    "full_auto")),
448 +             hook_summary=lambda: "",
449 +             mcp_summary=lambda: "",
450 +             plugin_summary=lambda: "",
451 +             cwd=str(tmp_path),
```

```
452 +             tool_registry=None,
453 +             app_state=None,
454 +             session_backend=None,
455 +             extra_skill_dirs=(),
456 +             extra_plugin_roots=(),
457 +         )
458 +
459 +     async def fake_start_runtime(bundle):
460 +         return None
461 +
462 +     monkeypatch.setattr("ohmo.gateway.runtime.build_runtime",
    fake_build_runtime)
463 +     monkeypatch.setattr("ohmo.gateway.runtime.start_runtime",
    fake_start_runtime)
464 +
465 +     with caplog.at_level(logging.WARNING):
466 +         pool = OhmoSessionRuntimePool(cwd=tmp_path, workspace=workspace,
    provider_profile="codex")
467 +         message = InboundMessage(channel="feishu", sender_id="u1",
    chat_id="c1", content="/permissions full_auto")
468 +         updates = [u async for u in pool.stream_message(message, "feishu:c1")]
469 +
470 +     assert handler_called is True
471 +     assert updates[-1].kind == "final"
472 +     assert updates[-1].text == "ran with full_auto"
473 +     assert "remote administrative command accepted" in caplog.text
474 +
475 +
359 476   @pytest.mark.asyncio
360 477   async def test_runtime_pool_includes_media_paths_in_prompt(tmp_path,
    monkeypatch):
361 478       workspace = tmp_path / ".ohmo-home"
```

Comments 0

Please sign in to comment.