volcengine / **OpenViking** Public

Code | Issues 96 | Pull requests 89 | Discussions | Actions | Projects

Commit c7bb167

Hinotoi-agent authored yesterday · 5 / 9 · Verified

```
[security] fix(bot): prevent unauthenticated remote bot control via OpenAPI HTTP routes (#1447)

* fix(bot): require API keys for OpenAPI HTTP routes

* refactor(bot): deduplicate bot channel auth checks

* fix(bot): format OpenAPI auth changes
```

main (#1447)

1 parent d13bf58 commit c7bb167

4 files changed +183 -30 lines changed

Top

Filter files…

- bot
  - tests
    - test_openapi_auth.py
  - vikingbot
    - channels
      - openapi.py
    - cli
      - commands.py
    - config
      - schema.py

4 files changed +183 -30 lines changed

Search within code

bot/tests/test_openapi_auth.py

```
@@ -0,0 +1,130 @@
+ # Copyright (c) 2026 Beijing Volcano Engine Technology Co., Ltd.
+ # SPDX-License-Identifier: AGPL-3.0
+ """Regression tests for OpenAPI HTTP auth requirements."""
+
+ import tempfile
+ from pathlib import Path
+
+ import pytest
+ from fastapi import FastAPI
+ from fastapi.testclient import TestClient
+ from vikingbot.bus.queue import MessageBus
+ from vikingbot.channels.openapi import OpenAPIChannel, OpenAPIChannelConfig
+ from vikingbot.channels.openapi_models import ChatResponse
+ from vikingbot.config.schema import BotChannelConfig
+
+
+ @pytest.fixture
+ def temp_workspace():
+     with tempfile.TemporaryDirectory() as tmpdir:
+         yield Path(tmpdir)
+
+
+ @pytest.fixture
+ def message_bus():
+     return MessageBus()
+
+
+ def _make_client(channel: OpenAPIChannel) -> TestClient:
+     app = FastAPI()
+     app.include_router(channel.get_router(), prefix="/bot/v1")
+     return TestClient(app)
+
+
+ class TestOpenAPIAuth:
+     def test_health_remains_available_without_api_key(self, message_bus,
  temp_workspace):
+         channel = OpenAPIChannel(
+             OpenAPIChannelConfig(api_key=""),
```

```
38  +             message_bus,
39  +             workspace_path=temp_workspace,
40  +         )
41  +         client = _make_client(channel)
42  +
43  +         response = client.get("/bot/v1/health")
44  +
45  +         assert response.status_code == 200
46  +
47  +     def test_chat_rejects_requests_when_api_key_not_configured(self,
      message_bus, temp_workspace):
48  +         channel = OpenAPIChannel(
49  +             OpenAPIChannelConfig(api_key=""),
50  +             message_bus,
51  +             workspace_path=temp_workspace,
52  +         )
53  +         client = _make_client(channel)
54  +
55  +         response = client.post("/bot/v1/chat", json={"message": "hello"})
56  +
57  +         assert response.status_code == 503
58  +         assert response.json()["detail"] == "OpenAPI channel API key is not
      configured"
59  +
60  +     def test_chat_accepts_request_with_configured_valid_api_key(
61  +         self, message_bus, temp_workspace, monkeypatch
62  +     ):
63  +         channel = OpenAPIChannel(
64  +             OpenAPIChannelConfig(api_key="secret123"),
65  +             message_bus,
66  +             workspace_path=temp_workspace,
67  +         )
68  +
69  +         async def fake_handle_chat(request):
70  +             return ChatResponse(
71  +                 session_id=request.session_id or "default", message="ok",
      events=None
72  +             )
73  +
74  +         monkeypatch.setattr(channel, "_handle_chat", fake_handle_chat)
```

```
75  +         client = _make_client(channel)
76  +
77  +         response = client.post(
78  +             "/bot/v1/chat",
79  +             headers={"X-API-Key": "secret123"},
80  +             json={"message": "hello"},
81  +         )
82  +
83  +         assert response.status_code == 200
84  +         assert response.json()["message"] == "ok"
85  +
86  +     def test_bot_channel_rejects_requests_when_channel_api_key_not_configured(
87  +         self, message_bus, temp_workspace
88  +     ):
89  +         channel = OpenAPIChannel(
90  +             OpenAPIChannelConfig(api_key="gateway-secret"),
91  +             message_bus,
92  +             workspace_path=temp_workspace,
93  +         )
94  +         channel._bot_configs["alpha"] = BotChannelConfig(id="alpha",
      api_key="")
95  +         client = _make_client(channel)
96  +
97  +         response = client.post(
98  +             "/bot/v1/chat/channel",
99  +             json={"message": "hello", "channel_id": "alpha"},
100 +         )
101 +
102 +         assert response.status_code == 503
103 +         assert response.json()["detail"] == "Bot channel 'alpha' API key is not
      configured"
104 +
105 +     def test_bot_channel_accepts_request_with_valid_api_key(
106 +         self, message_bus, temp_workspace, monkeypatch
107 +     ):
108 +         channel = OpenAPIChannel(
109 +             OpenAPIChannelConfig(api_key="gateway-secret"),
110 +             message_bus,
111 +             workspace_path=temp_workspace,
112 +         )
```

```
113 +         channel._bot_configs["alpha"] = BotChannelConfig(id="alpha", api_key="bot-secret")
114 +
115 +         async def fake_handle_bot_chat(channel_id, request):
116 +             return ChatResponse(
117 +                 session_id=request.session_id or "default", message=f"ok:{channel_id}"
118 +             )
119 +
120 +         monkeypatch.setattr(channel, "_handle_bot_chat", fake_handle_bot_chat)
121 +         client = _make_client(channel)
122 +
123 +         response = client.post(
124 +             "/bot/v1/chat/channel",
125 +             headers={"X-API-Key": "bot-secret"},
126 +             json={"message": "hello", "channel_id": "alpha"},
127 +         )
128 +
129 +         assert response.status_code == 200
130 +         assert response.json()["message"] == "ok:alpha"
```

bot/vikingbot/channels/openapi.py

```
@@ -28,9 +28,9 @@
28 28  )
29 29  from vikingbot.config.schema import (
30 30      BaseChannelConfig,
31 +     BotChannelConfig,
31 32      Config,
32 33      SessionKey,
33    -     BotChannelConfig,
34 34  )
35 35
36 36
@@ -179,10 +179,15 @@ async def send(self, msg: OutboundMessage) -> None:
179 179
180 180              pending = self._bot_pending[channel_id].get(session_id)
181 181              if not pending:
182 -                 logger.warning(f"No pending request for BotChannel {channel_id} session: {session_id}")
```

```
                +                 logger.warning(
                +                     f"No pending request for BotChannel {channel_id} session: {session_id}"
                +                 )
183 185                           return
184 186
185             -             if msg.event_type == OutboundEventType.RESPONSE or msg.event_type == OutboundEventType.NO_REPLY:
                +             if (
                +                 msg.event_type == OutboundEventType.RESPONSE
                +                 or msg.event_type == OutboundEventType.NO_REPLY
                +             ):
186 191                           await pending.add_event("response", msg.content or "")
187 192                           pending.set_final(msg.content or "")
188 193                           await pending.close_stream()
                  @@ -226,9 +231,12 @@ def _create_router(self) -> APIRouter:
226 231               channel = self  # Capture for closures
227 232
228 233               async def verify_api_key(x_api_key: Optional[str] = Header(None)) -> bool:
229             -             """Verify API key if configured."""
                +             """Verify API key for privileged HTTP chat/session routes."""
230 235                   if not channel.config.api_key:
231             -                 return True  # No auth required
                +                 raise HTTPException(
                +                     status_code=503,
                +                     detail="OpenAPI channel API key is not configured",
                +                 )
232 240                   if not x_api_key:
233 241                       raise HTTPException(status_code=401, detail="X-API-Key header required")
234 242                   # Use secrets.compare_digest for timing-safe comparison
                  @@ -332,10 +340,25 @@ async def delete_session(
332 340
333 341               # ========== Bot Channel Routes ==========
334 342
335             -         async def verify_bot_channel_api_key(x_api_key: Optional[str] = Header(None)) -> Optional[str]:
```

```
336      -             """Verify API key and return it if valid."""
         343 +         async def verify_bot_channel_api_key(
         344 +             x_api_key: Optional[str] = Header(None),
         345 +         ) -> Optional[str]:
         346 +             """Capture the raw bot-channel API key header for per-channel verification."""
337      347               return x_api_key
338      348
         349 +         def ensure_bot_channel_api_key(channel_id: str, x_api_key: Optional[str]) -> None:
         350 +             """Require an explicit per-channel API key for privileged bot HTTP routes."""
         351 +             bot_config = channel._bot_configs[channel_id]
         352 +             if not bot_config.api_key:
         353 +                 raise HTTPException(
         354 +                     status_code=503,
         355 +                     detail=f"Bot channel '{channel_id}' API key is not configured",
         356 +                 )
         357 +             if not x_api_key:
         358 +                 raise HTTPException(status_code=401, detail="X-API-Key header required")
         359 +             if not secrets.compare_digest(x_api_key, bot_config.api_key):
         360 +                 raise HTTPException(status_code=403, detail="Invalid API key")
         361 +
339      362           @router.post("/chat/channel", response_model=ChatResponse)
340      363           async def chat_channel(
341      364               request: ChatRequest,
         @@ -348,13 +371,7 @@ async def chat_channel(
348      371               if channel_id not in channel._bot_configs:
349      372                   raise HTTPException(status_code=404, detail=f"Channel '{channel_id}' not found")
350      373
351      -             # Verify API key for the specific channel
352      -             bot_config = channel._bot_configs[channel_id]
353      -             if bot_config.api_key:
354      -                 if not x_api_key:
355      -                     raise HTTPException(status_code=401, detail="X-API-Key header required")
356      -                 if not secrets.compare_digest(x_api_key, bot_config.api_key):
```

```
357      -                    raise HTTPException(status_code=403, detail="Invalid API
         key")
         374 +               ensure_bot_channel_api_key(channel_id, x_api_key)
358      375
359      376                 return await channel._handle_bot_chat(channel_id, request)
360      377
         @@ -370,13 +387,7 @@ async def chat_channel_stream(
370      387                 if channel_id not in channel._bot_configs:
371      388                     raise HTTPException(status_code=404, detail=f"Channel
         '{channel_id}' not found")
372      389
373      -                  # Verify API key for the specific channel
374      -                  bot_config = channel._bot_configs[channel_id]
375      -                  if bot_config.api_key:
376      -                      if not x_api_key:
377      -                          raise HTTPException(status_code=401, detail="X-API-Key
         header required")
378      -                      if not secrets.compare_digest(x_api_key, bot_config.api_key):
379      -                          raise HTTPException(status_code=403, detail="Invalid API
         key")
         390 +               ensure_bot_channel_api_key(channel_id, x_api_key)
380      391
381      392                 if not request.stream:
382      393                     request.stream = True
         @@ -609,7 +620,9 @@ async def _handle_bot_chat(self, channel_id: str,
         request: ChatRequest) -> ChatR
609      620                 if channel_id in self._bot_pending:
610      621                     self._bot_pending[channel_id].pop(session_id, None)
611      622
612      -      async def _handle_bot_chat_stream(self, channel_id: str, request:
         ChatRequest) -> StreamingResponse:
         623 +      async def _handle_bot_chat_stream(
         624 +          self, channel_id: str, request: ChatRequest
         625 +      ) -> StreamingResponse:
613      626                 """Handle a BotChannel streaming chat request."""
614      627                 session_id = request.session_id or str(uuid.uuid4())
615      628                 user_id = request.user_id or "anonymous"
         @@ -727,4 +740,4 @@ def get_openapi_router(bus: MessageBus, config: Config)
         -> APIRouter:
727      740                 )
```

```
728  741          logger.info(f"Subscribed to bot_api channel: {channel_id}")
729  742
730       -      return channel.get_router()
                ⊖
     743  +      return channel.get_router()
```

bot/vikingbot/cli/commands.py

```
     @@ -438,7 +438,7 @@ def prepare_channel(
438  438          openapi_config = OpenAPIChannelConfig(
439  439              enabled=True,
440  440              port=openapi_port,
441       -            api_key="",  # No auth required by default
     441  +            api_key="",
442  442          )
443  443          openapi_channel = OpenAPIChannel(
444  444              openapi_config,
     @@ -447,7 +447,9 @@ def prepare_channel(
447  447              global_config=config,
448  448          )
449  449          channels.add_channel(openapi_channel)
450       -        logger.info(f"OpenAPI channel enabled on port {openapi_port}")
     450  +        logger.info(
     451  +            f"OpenAPI channel enabled on port {openapi_port}; configure an API
          key before using HTTP chat endpoints"
     452  +        )
451  453
452  454      if channels.enabled_channels:
453  455          console.print(f"[green]✓[/green] Channels enabled: {',
          '.join(channels.enabled_channels)}")
```

bot/vikingbot/config/schema.py

```
     @@ -41,15 +41,18 @@ class SandboxMode(str, Enum):
41  41      SHARED = "shared"
42  42      PER_CHANNEL = "per-channel"
43  43
     44  +
44  45  class AgentMemoryMode(str, Enum):
45  46      """Agent memory mode enumeration."""
```

```
                47 +
46              48      PER_SESSION = "per-session"
47              49      SHARED = "shared"
48              50      PER_CHANNEL = "per-channel"
49              51
50              52
51              53  class BotMode(str, Enum):
52              54      """Bot running mode enumeration."""
                55 +
53              56      NORMAL = "normal"
54              57      READONLY = "readonly"
55              58      DEBUG = "debug"
@@ -119,7 +122,10 @@ class FeishuChannelConfig(BaseChannelConfig):
119             122     verification_token: str = ""
120             123     allow_from: list[str] = Field(default_factory=list)
121             124     allow_cmd_from: list[str] = Field(default_factory=list)  ## 允许执行命令的
                        Feishu用户ID列表
122             -       thread_require_mention: bool = Field(default=True, description="话题群模式下
                        是否需要@才响应：默认True=所有消息必须@才响应；False=新话题首条消息无需@，后续回复必须@")
                125 +   thread_require_mention: bool = Field(
                126 +       default=True,
                127 +       description="话题群模式下是否需要@才响应：默认True=所有消息必须@才响应；False=新
                        话题首条消息无需@，后续回复必须@",
                128 +   )
123             129
124             130     def channel_id(self) -> str:
125             131         # Use app_id directly as the ID
@@ -266,7 +272,7 @@ class OpenAPIChannelConfig(BaseChannelConfig):
266             272
267             273     type: ChannelType = ChannelType.OPENAPI
268             274     enabled: bool = True
269             -       api_key: str = ""  # If empty, no auth required
                275 +   api_key: str = ""  # Empty disables privileged HTTP routes until configured
270             276     allow_from: list[str] = Field(default_factory=list)
271             277     max_concurrent_requests: int = 100
272             278     _channel_id: str = "default"
@@ -280,7 +286,7 @@ class BotChannelConfig(BaseChannelConfig):
280             286
```

```
281 287         type: ChannelType = ChannelType.BOT_API
282 288         enabled: bool = True
283     -       api_key: str = ""  # If empty, no auth required
    289 +       api_key: str = ""  # Empty disables privileged HTTP routes until configured
284 290         allow_from: list[str] = Field(default_factory=list)
285 291         max_concurrent_requests: int = 100
286 292         need_mention: bool = False
    @@ -439,7 +445,9 @@ class ProviderConfig(BaseModel):
439 445
440 446         api_key: str = ""
441 447         api_base: Optional[str] = None
442     -       extra_headers: Optional[dict[str, str]] = Field(default_factory=dict)  # Custom headers (e.g. APP-Code for AiHubMix)
    448 +       extra_headers: Optional[dict[str, str]] = Field(
    449 +           default_factory=dict
    450 +       )  # Custom headers (e.g. APP-Code for AiHubMix)
443 451
444 452
445 453   class ProvidersConfig(BaseModel):
    @@ -803,4 +811,4 @@ def from_safe_name(safe_name: str):
803 811             file_name_split = safe_name.split("__")
804 812             return SessionKey(
805 813                 type=file_name_split[0], channel_id=file_name_split[1], chat_id=file_name_split[2]
806     -           )
    814 +           )
```

Comments 0

Please sign in to comment.