VetCoders / **mcp-server-semgrep** Public

Code | Issues | Pull requests | Discussions | Actions | Projects | Security and qu

2 Branches | 1 Tag | Go to file | Go to file | Code

Szowesgad Install Python/semgrep via uv; add README badge

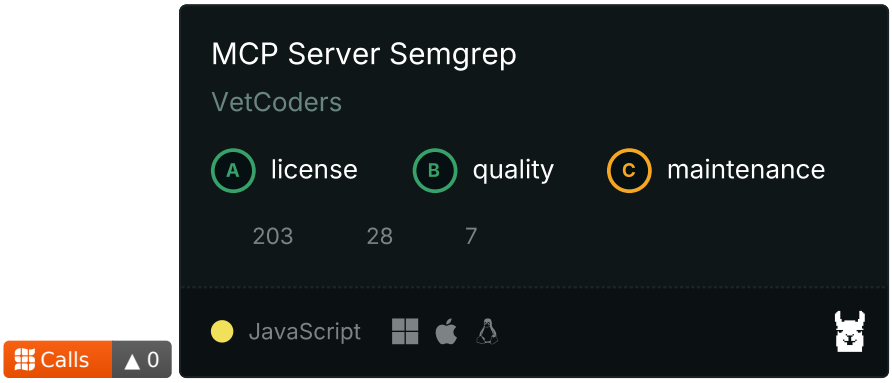
a8a727f · 3 days ago

.github	Detect and use Python in sem…	4 days ago
examples	Add fun and practical example …	last year
scripts	Providing registry ready solution	last week
src	(#15) fix(security): patch CWE-…	4 days ago
tests	(#15) fix(security): patch CWE-…	4 days ago
.dockerignore	Improve Dockerfile, Semgrep …	4 days ago
.gitignore	Improve Dockerfile, Semgrep …	4 days ago
.npmignore	Providing registry ready solution	last week
CHANGELOG.md	(#15) fix(security): patch CWE-…	4 days ago
CONTRIBUTING.md	Providing registry ready solution	last week
Dockerfile	Install Python/semgrep via uv; …	3 days ago
LICENSE	Organizing the final server and…	last year
README.md	Install Python/semgrep via uv; …	3 days ago
README_PL.md	(#15) fix(security): patch CWE-…	4 days ago
SECURITY.md	(#15) fix(security): patch CWE-…	4 days ago
USAGE.md	(#15) fix(security): patch CWE-…	4 days ago
eslint.config.js	(#15) fix(security): patch CWE-…	4 days ago
glama.json	Improve Dockerfile, Semgrep …	4 days ago
logo.svg	Add support for resources/list …	last year

package-lock.json	(#15) fix(security): patch CWE-…	4 days ago
package.json	(#15) fix(security): patch CWE-…	4 days ago
smithery.yaml	Add Smithery configuration	last week
test-rule.yaml	Add tests and example rule	last year
test-token.js	Fix Semgrep Pro token usage …	last week
tsconfig.json	Organizing the final server and…	last year
vitest.config.ts	Add tests and example rule	last year

README Contributing MIT license Security

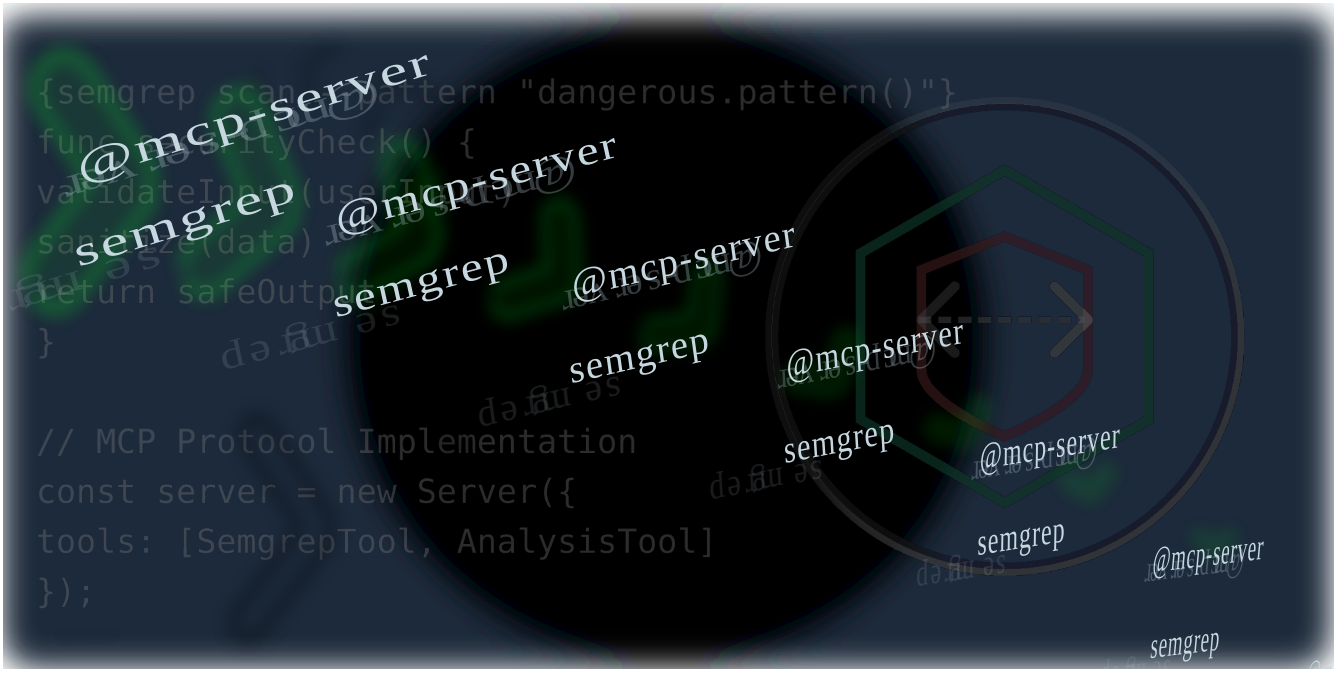
MCP Server Semgrep



POWERED BY:

Semgrep

About the Project



This project was initially inspired by robustness of Semgrep tool, The Replit Team and their Agent V2, as well as the implementation by stefanskiasan/semgrep-mcp-server, but has evolved with significant architectural changes for enhanced and easier installation and maintenance.

MCP Server Semgrep is a Model Context Protocol compliant server that integrates the powerful Semgrep static analysis tool with AI assistants like Anthropic Claude. It enables advanced code analysis, security vulnerability detection, and code quality improvements directly through a conversational interface.

Benefits of Integration

For Developers and Development Teams:

- **Holistic Source Code Analysis** - detecting issues throughout the entire project, not just in individual files
- **Proactive Error Detection** - identifying potential problems before they become critical bugs
- **Continuous Code Quality Improvement** - regular scanning and refactoring lead to gradual codebase improvements
- **Stylistic Consistency** - identification and fixing of inconsistencies in code, such as:
  - Arbitrary z-index layers in CSS
  - Inconsistent naming conventions
  - Code duplication
  - "Magic numbers" instead of named constants

For Security:

- **Automated Code Verification for Known Vulnerabilities** - scanning for known security issue patterns
- **Customized Security Rules** - creating project-specific rules
- **Team Education** - teaching secure programming practices through detection of potential issues

For Project Maintenance and Development:

- **"Live" Documentation** - AI can explain why a code fragment is problematic and how to fix it
- **Technical Debt Reduction** - systematically detecting and fixing problematic areas
- **Improved Code Reviews** - automatic detection of common issues allows focus on more complex matters

Key Features

- Direct integration with the official MCP SDK
- Simplified architecture with consolidated handlers
- Clean ES Modules implementation
- Efficient error handling and path validation for security
- Interface and documentation in both English and Polish
- Comprehensive unit tests
- Extensive documentation
- Cross-platform compatibility (Windows, macOS, Linux)
- Flexible Semgrep installation detection and management

Functions

Semgrep MCP Server provides the following tools:

- **scan_directory**: Scanning source code for potential issues
- **list_rules**: Displaying available rules and languages supported by Semgrep
- **analyze_results**: Detailed analysis of scan results
- **create_rule**: Creating custom Semgrep rules
- **filter_results**: Filtering results by various criteria
- **export_results**: Exporting results in various formats
- **compare_results**: Comparing two sets of results (e.g., before and after changes)

Common Use Cases

- Code security analysis before deployment
- Detection of common programming errors

- Enforcing coding standards within a team
- Refactoring and improving quality of existing code
- Identifying inconsistencies in styles and code structure (e.g., CSS, component organization)
- Developer education regarding best practices
- Verification of fix correctness (comparing before/after scans)

Installation

Prerequisites

- Node.js v18+
- TypeScript (for development)

Option 1: Install from Smithery.ai (Recommended)

The easiest way to install and use MCP Server Semgrep is through Smithery.ai:

1. Visit MCP Server Semgrep on Smithery.ai
2. Follow the installation instructions to add it to your MCP-compatible clients
3. Configure any optional settings like the Semgrep API token and allowed workspace roots

This is the recommended method for Claude Desktop and other MCP clients as it handles all dependencies and configuration automatically.

Option 2: Install from NPM Registry

```
# Using npm
npm install -g mcp-server-semgrep

# Using pnpm
pnpm add -g mcp-server-semgrep

# Using yarn
yarn global add mcp-server-semgrep
```

The package is also available on other registries:

- MCP.so

Option 3: Install from GitHub

```
# Using npm
npm install -g git+https://github.com/VetCoders/mcp-server-semgrep.git
```

```
# Using pnpm
pnpm add -g git+https://github.com/VetCoders/mcp-server-semgrep.git

# Using yarn
yarn global add git+https://github.com/VetCoders/mcp-server-semgrep.git
```

Option 4: Local Development Setup

1. Clone the repository:

```
git clone https://github.com/VetCoders/mcp-server-semgrep.git
cd mcp-server-semgrep
```

2. Install dependencies (supports all major package managers):

```
# Using pnpm (recommended)
pnpm install

# Using npm
npm install

# Using yarn
yarn install
```

3. Build the project:

```
# Using pnpm
pnpm run build

# Using npm
npm run build

# Using yarn
yarn build
```

> **Note**: The installation process will automatically check for Semgrep availability. If Semgrep is not found, you'll receive instructions on how to install it.

Workspace Root Contract

This server only reads and writes files inside explicitly allowed workspace roots.

- By default, the allowed root is the process working directory (`process.cwd()`).
- For Claude Desktop, Smithery, or any launcher that does not start the server inside your project root, set `MCP_SERVER_SEMGREP_ALLOWED_ROOTS` to one or more absolute directories.

- Use your platform path delimiter for multiple roots: `:` on macOS/Linux, `;` on Windows.

Authentication Modes

This server does not implement its own Semgrep account handling. It shells out to the installed `semgrep` CLI and relies on Semgrep's normal authentication behavior.

- Local terminal and local development runs can often use an existing `semgrep login` session from the current OS account.
- Managed launches such as Claude Desktop, Smithery, containers, or CI should prefer an explicit `SEMGREP_APP_TOKEN` for deterministic behavior.
- `SEMGREP_APP_TOKEN` is still the safest option when you need portable configuration across machines or runners.

Semgrep Installation Options

Semgrep can be installed in several ways:

- **Via package managers**:

```
# Using pnpm
pnpm add -g semgrep

# Using npm
npm install -g semgrep

# Using yarn
yarn global add semgrep
```

- **Python pip**:

```
pip install semgrep
```

- **Homebrew** (macOS):

```
brew install semgrep
```

- **Linux**:

```
sudo apt-get install semgrep
# or
curl -sSL https://install.semgrep.dev | sh
```

- **Windows**:

```
pip install semgrep
```

Integration with Claude Desktop

There are two ways to integrate MCP Server Semgrep with Claude Desktop:

Method 1: Install via Smithery.ai (Recommended)

1. Visit MCP Server Semgrep on Smithery.ai
2. Click "Install in Claude Desktop"
3. Follow the on-screen instructions

Method 2: Manual Configuration

1. Install Claude Desktop
2. Update the Claude Desktop configuration file (`claude_desktop_config.json`) and add this to your servers section.

For local launches started under a user account that is already authenticated with `semgrep login`, the Semgrep CLI may be able to reuse that login. For desktop-managed or shared environments, we still recommend setting `SEMGREP_APP_TOKEN` explicitly:

```
{
  "mcpServers": {
    "semgrep": {
      "command": "node",
      "args": [
        "/your_path/mcp-server-semgrep/build/index.js"
      ],
      "env": {
        "SEMGREP_APP_TOKEN": "your_semgrep_app_token",
        "MCP_SERVER_SEMGREP_ALLOWED_ROOTS": "/Users/you/projects"
      }
    }
  }
}
```

3. Launch Claude Desktop and start asking questions about code analysis.

If you want to scan more than one workspace, set `MCP_SERVER_SEMGREP_ALLOWED_ROOTS` to a platform-delimited list of absolute paths.

Usage Examples

Project Scanning

```
Could you scan my source code in the /projects/my-application directory for potential security issues? That directory is already included in MCP_SERVER_SEMGREP_ALLOWED_ROOTS.
```

Style Consistency Analysis

```
Analyze the z-index values in the project's CSS files and identify inconsistencies and potential layer conflicts.
```

Creating a Custom Rule

```
Create a Semgrep rule that detects improper use of input sanitization functions.
```

Filtering Results

```
Show me only scan results related to SQL injection vulnerabilities.
```

Identifying Problematic Patterns

```
Find all "magic numbers" in the code and suggest replacing them with named constants.
```

Creating Custom Rules

You can create custom rules for your project's specific needs. Here are examples of rules you can create:

Rule to detect inconsistent z-indices:

```yaml
rules:
  - id: inconsistent-z-index
    pattern: z-index: $Z
    message: "Z-index $Z may not comply with the project's layering system"
```

```
    languages: [css, scss]
    severity: WARNING
```

Rule to detect deprecated imports:

```
rules:
  - id: deprecated-import
    pattern: import $X from 'old-library'
    message: "You're using a deprecated library. Consider using 'new-library'"
    languages: [javascript, typescript]
    severity: WARNING
```

Development

Testing

```
pnpm test
```

Project Structure

```
├── src/
│   └── index.ts          # Main entry point and all handler implementations
├── scripts/
│   └── check-semgrep.js  # Semgrep detection and installation helper
├── build/                # Compiled JavaScript (after build)
└── tests/                # Unit tests
```

Further Documentation

Detailed information on using the tool can be found in:

- USAGE.md - Detailed usage instructions
- README_PL.md - Documentation in Polish
- examples/ - Example fun and practical Semgrep rules - "The Hall of Code Horrors"

License

This project is licensed under the MIT License - see the LICENSE file for details.

Releases 1

v1.0.1 Latest
yesterday

Packages

No packages published

Contributors 8

Languages

JavaScript 75.4% **TypeScript** 22.5% **Dockerfile** 2.1%