software-checker-mcpリポジトリセキュリティ分析レポート

📝 1. 基本情報評価

- リポジトリ名/URL：software-checker-mcp
- 開発者/組織：不明（リポジトリから直接確認できず）
- 最終更新日と更新頻度：最終更新は2024年4月8日（最近更新されたファイル: main.py）
- GitHub統計：リポジトリからは直接確認できず
- イシュー対応状況：リポジトリからは直接確認できず
- ライセンス：明示的に記載なし

📑 2. コードベース概要

- 使用言語とフレームワーク：Python（3.12以上）、FastMCP（MCPサーバー用フレームワーク）
- 主要ディレクトリ構造：

```
software-checker-mcp/
├── assets/
├── src/
│   └── software_checker_mcp/
│       ├── __init__.py
│       ├── main.py
│       ├── git-release-notes-generator-prompt_v1.md
│       └── repo-review-prompt-v3.md
├── pyproject.toml
├── README.md
└── .gitignore
```

- 依存ライブラリとそのバージョン：
  - fastmcp>=0.1.0,<0.2.0
  - uvicorn>=0.15.0,<0.16.0
- ビルド/テストシステム：hatchling（ビルドシステム）

🔧 3. ツール定義・説明文分析

対象ファイル：src/software_checker_mcp/main.py

ツール定義方法の概要

このMCPは2つのツールを定義しています：

1. `repository_quality_check`：リポジトリの品質を確認するプロンプトを提供
2. `generate_release_notes`：Gitリリースノートを生成するプロンプトを提供

両方のツールとも、外部のマークダウンファイルからプロンプトテンプレートを読み込み、LLMモデルに送信するための指示とともに返します。

各ツールの説明文分析

repository_quality_check

python

```python
@mcp.tool()
def repository_quality_check(llm_model: str = "Claude 3.7 Sonnet") -> dict:
    """
    リポジトリ品質確認プロンプトを返す

    Args:
        llm_model: 使用するLLMモデル名 (デフォルト: "Claude 3.7 Sonnet")

    Returns:
        プロンプトと指示を含む辞書
    """
```

- 隠しメッセージや二重の指示：なし

- AIモデルへの直接的な誘導：あり（特定のLLMモデルの名前がデフォルト値として設定されているが悪意はない）
- 機密情報へのアクセス要求：なし
- 悪意のあるコードの埋め込み：なし

generate_release_notes

```python
@mcp.tool()
def generate_release_notes(
    current_tag: str = None,
    previous_tag: str = None,
    llm_model: str = "Claude 3.7 Sonnet"
) -> dict:
    """
    Gitリリースノート生成プロンプトを返す

    Args:
        current_tag: 現在のタグ（指定しない場合は最新のタグ）
        previous_tag: 前回のタグ（指定しない場合は現在のタグの一つ前）
        llm_model: 使用するLLMモデル名 (デフォルト: "Claude 3.7 Sonnet")

    Returns:
        プロンプトと指示を含む辞書
    """
```

- 隠しメッセージや二重の指示：なし
- AIモデルへの直接的な誘導：あり（特定のLLMモデルの名前がデフォルト値として設定されているが悪意はない）
- 機密情報へのアクセス要求：なし
- 悪意のあるコードの埋め込み：なし

ツール説明文の動的生成や更新機能の有無

- 説明文自体の動的生成はないが、プロンプトの内容は外部ファイルから読み込まれるため、それらのファイルを変更することでツールの動作を変更できる

⌨ 4. コマンド実行とユーザー入力処理

対象ファイル：src/software_checker_mcp/main.py

危険な関数/メソッドの使用状況

- `subprocess.run()`: Gitコマンドを実行するために使用されているが、制限された形式で使用されている
- `eval()`, `Function()`, `exec()`: 使用されていない

ユーザー入力のバリデーション実装

- `validate_git_tag()` 関数が実装されており、Gitタグのフォーマットをチェックしている

```python
def validate_git_tag(tag: str) -> bool:
    # 安全なGitタグフォーマットのみを許可
    import re
    return bool(re.match(r'^[a-zA-Z0-9_\.\-]+$', tag))
```

- 英数字、アンダースコア、ドット、ハイフンのみを許可する正規表現でフィルタリングされている

シェルコマンド構築方法

- 固定のコマンド引数配列を使用（文字列連結なし）

```python
result = subprocess.run(
    ["git", "tag", "--sort=-version:refname"],
    capture_output=True,
    text=True,
    check=True,
    timeout=5
)
```

コマンドインジェクション対策の評価

- 適切な対策が実装されている：
  - `subprocess.run()` に配列形式でコマンドを渡し、シェル経由での実行を避けている
  - ユーザー入力（タグ）を検証関数で確認している
  - タイムアウト設定がある（5秒）

🔐 5. 認証情報・トークン管理

対象ファイル：src/software_checker_mcp/main.py

- 認証情報は保存・管理されていない
- トークン管理の実装はない
- このMCPでは認証を必要とする外部サービスにアクセスしていないため、認証情報の保護は不要

📂 6. ファイルシステムアクセス

対象ファイル：src/software_checker_mcp/main.py

アクセス可能なファイルとディレクトリ

- 現在のディレクトリ内の特定のマークダウンファイル（`repo-review-prompt-v3.md` と `git-release-notes-generator-prompt_v1.md`）のみにアクセス

```
python

 current_dir = Path(__file__).parent
repo_review_path = current_dir / "repo-review-prompt-v3.md"
```

センシティブファイルへのアクセス可能性

- MCPはホームディレクトリやシステムファイルにアクセスしていない
- アクセスはモジュール自身のディレクトリ内のファイルに限定されている

ファイルパス構築の安全性

- `Path` オブジェクトを使用して安全にパスを構築
- パストラバーサル攻撃の可能性は低い（ユーザー入力からパスを構築していない）

📡 7. 通信とデータ送信

対象ファイル：src/software_checker_mcp/main.py

- MCPは外部サーバーとの通信を行っていない
- すべての処理はローカル環境内で完結する
- データ送信を行うコードは存在しない

☣️ 8. 複数サーバー連携時のセキュリティ

対象ファイル：src/software_checker_mcp/main.py

- 複数サーバー連携の実装はない
- ツール衝突解決メカニズムは実装されていない（単一のMCPサーバーのみ）
- シャドーイング攻撃対策は必要ない（他のサーバーとの連携がない）

🔄 9. アップデートとバージョン管理

対象ファイル：src/software_checker_mcp/main.py, pyproject.toml

- ツール定義の更新方法は明示的に実装されていない
- バージョン情報は `pyproject.toml` に記載されている（0.1.0）
- プロンプトテンプレートのファイル名にバージョン番号が含まれている（`repo-review-prompt-v3.md`, `git-release-notes-generator-prompt_v1.md`）

!? 10. 不審なコードパターンの検出

難読化されたコード

- Base64エンコード文字列: なし
- 意図的に複雑化された論理: なし

隠しバックドア

- 不審なコメントアウトコード: なし
- 条件分岐による隠し機能: なし

データ流出の可能性

- 機密情報の収集コード: なし
- ロギング実装: 明示的なロギング実装はない（エラー時の出力のみ）

📊 11. 総合的なセキュリティ評価

発見された主要な脆弱性と深刻度

- 重大な脆弱性は発見されなかった
- ユーザー入力（Gitタグ）の検証が適切に行われている
- シェルコマンド実行も安全な方法で実装されている

攻撃ベクトルの可能性

- プロンプトファイルの内容を変更することで、LLMへの指示を操作できる可能性（低リスク）
- ファイルシステムアクセスは限定的で安全

実装されているセキュリティ対策

- ユーザー入力（Gitタグ）の正規表現による検証
- 安全なサブプロセス実行（配列形式のコマンド指定）
- タイムアウト設定による実行時間の制限

使用判断

- 安全に使用可能 😄
  - コードは比較的シンプルで、危険な操作を行っていない
  - ユーザー入力の検証が適切に実装されている
  - すべての処理がローカル環境内で完結する

🪄 12. 改善提案

発見された問題に対する修正提案

1. エラーハンドリングの強化
   - すべての例外をより詳細にキャッチし、適切なエラーメッセージを返すように改善

```
python
```

```python
try:
    # 処理
except FileNotFoundError as e:
    # ファイル関連のエラー処理
except subprocess.SubprocessError as e:
    # プロセス関連のエラー処理
except Exception as e:
    # その他の例外処理
```

追加すべきセキュリティ対策

1. プロンプトファイルの完全性チェック
   - ファイルのハッシュ値を検証して、不正な変更がないことを確認

```
python
```

```python
import hashlib
def validate_file_integrity(file_path, expected_hash):
    with open(file_path, 'rb') as f:
        file_hash = hashlib.sha256(f.read()).hexdigest()
    return file_hash == expected_hash
```

2. ロギング機能の追加
   - 重要な操作やエラーを記録するためのロギング機能

```python
import logging
logging.basicConfig(level=logging.INFO, filename='mcp.log')
logging.info('MCP Server started')
```

優先的に対応すべき脆弱性

- 緊急に対応すべき深刻な脆弱性は見つかりませんでした
- 改善提案は優先度の低い強化策です

追加情報

README の情報を元に、分析レポートに以下の追加情報を記載します：

基本情報の更新

- リポジトリURL：https://github.com/Tomatio13/software-checker-mcp.git（READMEのインストール手順から）
- 開発者/組織：Tomatio13（GitHub）、元のプロンプト作成者はMakiさん（Sunwood-ai-lbas）

コードベース概要の補足

- ディレクトリ構造（READMEより更新）：

```
software-checker-mcp/
├── pyproject.toml        # プロジェクト設定ファイル
├── README.md             # READMEファイル
├── README_JP.md          # 日本語版README
├── start.sh              # 起動スクリプト
├── assets/               # 画像などのアセット
│   ├── quality_check_sample.png
│   └── release_note_sample.png
└── src/
    └── software_checker_mcp/
        ├── __init__.py               # パッケージ初期化ファイル
        ├── main.py                   # MCPサーバーのメイン実装
        ├── repo-review-prompt-v3.md  # リポジトリ品質確認用プロンプト
        └── git-release-notes-generator-prompt_v1.md  # リリースノート生成用プロン
```

プロンプトファイルの出典

README の情報から、プロンプトファイルの出典が明らかになりました：

- リポジトリ品質確認プロンプト V3とGitリリースノート作成プロンプト V1は、Makiさん（Sunwood-ai-lbas）が公開しているプロンプトをベースにしています
- 出典リポジトリ：https://github.com/Sunwood-ai-labs/MysticLibrary/tree/main/prompts/coding

利用方法の補足

- MCPサーバーはClaude Desktopと連携するための設定例が提供されています
- 必要な実行環境は、Python 3.12以上とuvパッケージマネージャーです

これらの追加情報を考慮すると、セキュリティ評価に大きな変更はありませんが、プロンプトファイルが外部ソースから採用されている点は注目に値します。ただし、それらのプロンプト自体には悪意のある内容は含まれていません。

分析結果の要約

本レポートでは、software-checker-mcpのソースコードをセキュリティリスクの観点から包括的に分析しました。

主な調査結果

1. **低リスク評価**: このMCPサーバーは全体的に安全に設計されており、セキュリティリスクは極めて低いと評価できます。
2. **限定的な機能**: 機能はリポジトリ品質チェックとリリースノート生成の2つの特定タスクに限定されています。
3. **安全な実装**:

    - ユーザー入力（Gitタグ）の適切な検証が行われています
    - コマンド実行は安全な方法で実装されています
    - ファイルアクセスは限定的で、センシティブな場所にはアクセスしません
4. **外部通信なし**: 外部サーバーとの通信がなく、すべての処理がローカル環境内で完結します

セキュリティ観点での強み

- 厳格な入力検証メカニズム
- シェルインジェクションを防ぐ安全なサブプロセス実行
- 秘密情報や機密データの取り扱いなし
- タイムアウト設定による実行時間の制限

改善可能な点

- より詳細なエラーハンドリングとロギング機能
- プロンプトファイルの完全性検証メカニズム
- 明示的なライセンス情報の追加

最終評価

software-checker-mcpは、セキュリティリスクが低く、意図した目的（リポジトリ品質チェックとリリースノート生成）に安全に使用できるMCPサーバーと評価します。悪意のあるコードやバックドアは発見されず、ユーザーのプライバシーやデータセキュリティを脅かす要素も見つかりませんでした。

使用判断: **安全に使用可能** 😄