Command Injection in Bash Automation Step

Critical **mjashanks** published **GHSA-gjw9-34gf-rp6m** 4 days ago

Package

No package listed

Affected versions

<= 3.23.22

Patched versions

3.33.4

Description

Location: `packages/server/src/automations/steps/bash.ts`

Description

The bash automation step executes user-provided commands using `execSync` without proper sanitization or validation. User input is processed through `processStringSync` which allows template interpolation, potentially allowing arbitrary command execution.

Code Reference

```
const command = processStringSync(inputs.code, context)

let stdout,
  success = true
try {
  stdout = execSync(command, {
    timeout: environment.QUERY_THREAD_TIMEOUT,
  }).toString()
```

Attack Vector

An attacker with access to create or modify automations can inject malicious shell commands by including template syntax that evaluates to command injection payloads (e.g., `$(rm -rf /)`, `; malicious-command`, `| malicious-command`).

Impact

- Remote code execution (RCE)
- Complete system compromise
- Data exfiltration
- Lateral movement within the infrastructure

Recommendation

1. **Immediate**: Disable bash automation step in production until fixed
2. Implement a whitelist of allowed commands
3. Use parameterized command execution with proper escaping
4. Implement command argument validation
5. Consider using a restricted shell or command sandboxing
6. Add rate limiting and monitoring for command execution

Example Fix

```
import { spawn } from "child_process"

// Validate against whitelist
const ALLOWED_COMMANDS = ["echo", "date", "pwd"] // Extend as needed

function sanitizeCommand(input: string): string {
  // Remove dangerous characters and command chaining
  return input.replace(/[;&|`$(){}[\]]/g, "").trim()
}

function validateCommand(cmd: string): boolean {
  const parts = cmd.split(/\s+/)
  return ALLOWED_COMMANDS.includes(parts[0])
}

export async function run({ inputs, context }) {
  if (!inputs.code) {
    return { stdout: "Budibase bash automation failed: Invalid inputs" }
  }

  const processedCommand = processStringSync(inputs.code, context)
  const sanitized = sanitizeCommand(processedCommand)

  if (!validateCommand(sanitized)) {
    return {
      success: false,
      stdout: "Command not allowed"
    }
  }

  // Use spawn instead of execSync with proper argument handling
  return new Promise((resolve) => {
    const [command, ...args] = sanitized.split(/\s+/)
```

```
    const proc = spawn(command, args, {
      timeout: environment.QUERY_THREAD_TIMEOUT,
    })

    let stdout = ""
    proc.stdout.on("data", (data) => { stdout += data })
    proc.on("close", (code) => {
      resolve({ stdout, success: code === 0 })
    })
  })
}
```

Severity

Critical

CVE ID

CVE-2026-25044

Weaknesses

No CWEs

Credits

 **omkarparth** Reporter