AI Tools Security Guide

Security Architecture

The AI Tools framework implements defense-in-depth security with multiple layers of protection.

Security Layers

1. Input Validation

All inputs are validated before execution:

```
// Path validation
- Resolve to absolute paths
- Check against allowed base paths
- Block access to sensitive directories
- Validate file extensions

// Command validation
- Parse command structure
- Check against whitelist/blacklist
- Detect dangerous patterns
- Sanitize arguments
```

2. Sandboxing

Code execution is isolated:

```
// Python execution
- Spawned in separate process
- No access to parent environment
- Limited system calls
- Timeout enforcement

// JavaScript execution
- VM2 sandbox
- No access to Node.js APIs
- No file system access
- No network access
```

3. Resource Limits

All operations have resource constraints:

```
{
  maxExecutionTime: 30000,    // 30 seconds
  maxMemoryMB: 512,           // 512 MB
  maxFileSize: 10485760,      // 10 MB
  maxConcurrentOps: 5         // 5 operations
}
```

4. Audit Logging

All operations are logged:

```
{
  timestamp: "2024-01-01T12:00:00Z",
  type: "execution",
  toolName: "read_file",
  parameters: { path: "/path/to/file" },
  userId: "user123",
  success: true,
  executionTime: 45
}
```

Threat Model

Threats Addressed

1. **Unauthorized File Access**
   - Mitigation: Path validation and whitelist
   - Status: ✅ Protected

2. **Code Injection**
   - Mitigation: Input sanitization and sandboxing
   - Status: ✅ Protected

3. **Resource Exhaustion**
   - Mitigation: Timeouts and memory limits
   - Status: ✅ Protected

4. **Privilege Escalation**
   - Mitigation: No sudo/root access, command whitelist
   - Status: ✅ Protected

5. **Data Exfiltration**
   - Mitigation: No network access in sandbox
   - Status: ✅ Protected

Residual Risks

1. **AI Prompt Injection**
   - Risk: Malicious prompts could trick AI
   - Mitigation: User review of operations
   - Status: ⚠️ Requires user vigilance

2. **Logic Bugs**
   - Risk: Bugs in validation logic
   - Mitigation: Code review and testing
   - Status: ⚠️ Ongoing monitoring

Security Configuration

Recommended Settings

Production Environment

```json
{
  "security": {
    "filesystem": {
      "allowedBasePaths": ["./src", "./public"],
      "blockedPaths": [
        "./node_modules",
        "./.git",
        "./.env",
        "./prisma"
      ],
      "maxFileSizeMB": 5
    },
    "codeExecution": {
      "allowedLanguages": ["javascript"],
      "maxExecutionTimeMs": 15000,
      "maxMemoryMB": 256,
      "allowNetworkAccess": false
    },
    "general": {
      "requireApprovalForDangerous": true
    }
  }
}
```

Development Environment

```json
{
  "security": {
    "filesystem": {
      "allowedBasePaths": ["./"],
      "maxFileSizeMB": 10
    },
    "codeExecution": {
      "allowedLanguages": ["python", "javascript", "bash"],
      "maxExecutionTimeMs": 30000,
      "maxMemoryMB": 512
    }
  }
}
```

Security Checklist

Before Deployment

- [ ] Review and update security configuration
- [ ] Test all security boundaries
- [ ] Enable audit logging
- [ ] Set up log monitoring
- [ ] Configure alerts for suspicious activity
- [ ] Document security procedures

- [ ] Train users on security best practices
- [ ] Set up regular security audits

Regular Maintenance

- [ ] Review audit logs weekly
- [ ] Update command whitelist as needed
- [ ] Patch security vulnerabilities
- [ ] Review and rotate access credentials
- [ ] Test disaster recovery procedures
- [ ] Update security documentation

Incident Response

If Security Breach Detected

1. **Immediate Actions**
   - Disable AI tools immediately
   - Preserve logs for analysis
   - Notify security team
   - Document the incident

2. **Investigation**
   - Review audit logs
   - Identify attack vector
   - Assess damage
   - Collect evidence

3. **Remediation**
   - Patch vulnerabilities
   - Update security rules
   - Reset compromised credentials
   - Restore from backup if needed

4. **Post-Incident**
   - Conduct post-mortem
   - Update security procedures
   - Implement additional controls
   - Train team on lessons learned

Security Testing

Manual Testing

```bash
# Test path traversal protection
curl -X POST /api/ai/tool-chat \
  -d '{"message": "Read file ../../etc/passwd"}'

# Test command injection
curl -X POST /api/ai/tool-chat \
  -d '{"message": "Run command: ls; rm -rf /"}'

# Test resource limits
curl -X POST /api/ai/tool-chat \
  -d '{"message": "Run infinite loop"}'
```

Automated Testing

```typescript
// security.test.ts
describe('Security Tests', () => {
  test('blocks path traversal', async () => {
    const result = await executeTool('read_file', {
      path: '../../etc/passwd'
    });
    expect(result.success).toBe(false);
  });

  test('blocks dangerous commands', async () => {
    const result = await executeTool('execute_shell', {
      command: 'rm -rf /'
    });
    expect(result.success).toBe(false);
  });

  test('enforces timeouts', async () => {
    const result = await executeTool('execute_python', {
      code: 'while True: pass',
      timeout: 1000
    });
    expect(result.success).toBe(false);
    expect(result.error).toContain('timeout');
  });
});
```

Compliance

Data Protection

- All file operations are logged
- User data is not transmitted externally
- Logs contain no sensitive information
- Data retention policies are enforced

Access Control

- Role-based access control (RBAC)
- Principle of least privilege

- Regular access reviews
- Audit trail for all operations

Security Updates

Keeping Secure

1. **Dependencies**
   ```
   bash
      npm audit
      npm audit fix
   ```

2. **Security Patches**
   - Monitor security advisories
   - Apply patches promptly
   - Test after updates

3. **Configuration**
   - Review quarterly
   - Update based on threats
   - Document changes

Contact

For security concerns:
- Email: security@example.com
- Emergency: +1-XXX-XXX-XXXX
- Bug Bounty: https://example.com/security

References

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- CWE Top 25 (https://cwe.mitre.org/top25/)
- NIST Cybersecurity Framework (https://www.nist.gov/cyberframework)