🔒 Security Policy

📊 Supported Versions

We actively maintain and provide security updates for the following versions:

Version	Supported	Status
1.0.x	✅ Yes	Current stable release
0.9.x	✅ Yes	Previous stable (until 2025-12-01)
< 0.9	❌ No	End of life

🚨 Reporting a Vulnerability

We take security seriously. If you discover a security vulnerability, please follow these steps:

🔒 Private Disclosure Process

1. **Do NOT** create a public GitHub issue for security vulnerabilities
2. **Email** security concerns to: `security@pipeline-automation-hub.dev` (if available) or create a private issue
3. **Include** detailed information about the vulnerability:
   - Type of vulnerability
   - Steps to reproduce
   - Potential impact
   - Suggested fix (if known)

📋 What to Include

Please provide as much information as possible:

- **Description**: Clear description of the vulnerability
- **Location**: File paths, line numbers, affected components
- **Reproduction**: Step-by-step instructions to reproduce
- **Impact**: How this could be exploited and potential damage
- **Environment**: Versions, operating system, browser (if applicable)
- **Proof of Concept**: Code snippet or screenshots (if safe to share)

🛡️ Security Measures

Document Processing Security

- **File Validation**: All uploaded files are validated before processing
- **Sanitization**: Content is sanitized during processing
- **Isolation**: Processing runs in controlled environment

- **Size Limits**: File size restrictions prevent resource exhaustion
- **Type Checking**: Only supported file types are processed

Web Application Security

- **Authentication**: NextAuth.js for secure authentication
- **Authorization**: Role-based access control (when implemented)
- **CSRF Protection**: Built-in CSRF protection
- **Input Validation**: All user inputs validated and sanitized
- **Headers**: Security headers implemented
- **HTTPS**: Enforce HTTPS in production

API Security

- **Rate Limiting**: API endpoints have rate limiting
- **Input Validation**: All API inputs validated
- **Error Handling**: Secure error messages (no sensitive data leak)
- **Authentication**: API endpoints require proper authentication
- **Logging**: Security events are logged

Infrastructure Security

- **Dependencies**: Regular dependency updates
- **Secrets Management**: Environment variables for sensitive data
- **Access Control**: Principle of least privilege
- **Monitoring**: Security monitoring in place
- **Backups**: Regular backups with encryption

🔍 Security Best Practices

For Contributors

- **Code Review**: All code goes through security-focused reviews
- **Dependencies**: Only trusted dependencies are added
- **Secrets**: Never commit secrets, API keys, or credentials
- **Testing**: Security testing for new features
- **Documentation**: Security implications documented

For Users

- **Updates**: Keep the application updated to latest version
- **Environment**: Use secure environment variables
- **Access**: Limit access to authorized users only
- **Files**: Only process trusted document files
- **Monitoring**: Monitor for unusual activity

For Administrators

- **Deployment**: Use secure deployment practices
- **Monitoring**: Implement comprehensive monitoring
- **Backups**: Regular secure backups
- **Access**: Restrict administrative access
- **Logs**: Review security logs regularly

🚨 Known Security Considerations

Document Processing

- **File Upload**: Only process files from trusted sources
- **Content**: Be aware that processed content reflects input documents
- **Metadata**: Metadata extraction may include sensitive information
- **Storage**: Processed files are stored according to configured policies

GitHub Integration

- **Token Security**: GitHub tokens should have minimal required permissions
- **Repository Access**: Limit repository access to necessary users
- **Webhook Security**: Secure webhook endpoints (if implemented)

Database Security (if using database features)

- **Connection**: Use encrypted database connections
- **Access**: Database access restricted to application
- **Backup**: Secure database backups
- **Migration**: Review database migrations for security implications

🛠️ Security Tools and Automation

Automated Security Checks

- **Dependency Scanning**: Automated vulnerability scanning
- **Code Analysis**: Static code analysis for security issues
- **License Compliance**: License compatibility checking
- **Secret Scanning**: Prevent secrets from being committed

CI/CD Security

- **Build Security**: Secure build environment
- **Test Security**: Security tests in CI pipeline
- **Deployment**: Secure deployment processes
- **Monitoring**: Continuous security monitoring

📈 Security Updates

Update Process

1. **Assessment**: Evaluate security impact
2. **Fix**: Develop and test security fix
3. **Release**: Release security update
4. **Communication**: Notify users of security updates
5. **Monitoring**: Monitor for successful deployment

Communication Channels

- **GitHub Security Advisories**: For public vulnerabilities
- **Release Notes**: Security fixes noted in releases
- **Documentation**: Security updates in documentation

- **Email**: Direct notification for critical issues (if contact available)

🎯 Response Timeline

Initial Response

- **24 hours**: Acknowledge receipt of security report
- **48 hours**: Initial assessment and response plan
- **7 days**: Detailed response with timeline for fix

Resolution Timeline

- **Critical**: 24-48 hours for patch release
- **High**: 7 days for patch release
- **Medium**: 30 days for next planned release
- **Low**: Next major release or as appropriate

📚 Security Resources

Documentation

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- Next.js Security (https://nextjs.org/docs/advanced-features/security-headers)
- Node.js Security (https://nodejs.org/en/docs/guides/security/)
- GitHub Security (https://docs.github.com/en/code-security)

Security Scanning Tools

- `npm audit` / `yarn audit` for dependency vulnerabilities
- ESLint security rules for code analysis
- GitHub Security Advisories for public vulnerabilities
- Dependabot for automated dependency updates

🔐 Encryption and Data Protection

Data at Rest

- **Files**: Processed files stored securely
- **Metadata**: Metadata encrypted when sensitive
- **Secrets**: Environment variables and secrets encrypted
- **Backups**: Backup encryption enabled

Data in Transit

- **HTTPS**: All web traffic encrypted
- **API**: API calls encrypted in transit
- **Database**: Database connections encrypted
- **File Transfer**: Secure file transfer protocols

📞 Contact Information

For security-related questions or concerns:

- **Security Issues**: Create a private security advisory on GitHub
- **General Questions**: Open a public issue (non-security related only)
- **Documentation**: Contribute security improvements via pull request

⚖️ Responsible Disclosure

We are committed to working with security researchers and the community to verify, reproduce, and respond to legitimate reported vulnerabilities. We ask that you:

- **Give us time** to investigate and fix the issue before public disclosure
- **Avoid** accessing, modifying, or deleting data during testing
- **Don’t** perform attacks that could harm service availability
- **Don’t** access other users’ data or violate privacy
- **Follow** applicable laws and regulations

🏆 Security Recognition

We appreciate security researchers and contributors who help improve the security of Pipeline Automation Hub. Contributors to security improvements may be recognized:

- **Hall of Fame**: Security contributors listed in project documentation
- **Release Notes**: Security fix contributors acknowledged
- **Community Recognition**: Public thanks for responsible disclosure

---

This security policy is part of the Pipeline Automation Hub project and is updated regularly to reflect current security practices and procedures.

Last Updated: September 2025
Version: 1.0.0