🔒 Security Policy

Supported Versions

We actively support the following versions with security updates:

Version	Supported
1.x.x	✅ Yes
< 1.0	❌ No

🛡️ Security Features

Authentication & Authorization

- **NextAuth.js** integration with secure session management
- **Role-based access control** (Admin/Participant)
- **Configurable authentication** (authenticated vs anonymous modes)
- **Session timeout** and automatic logout
- **CSRF protection** built into NextAuth

Data Protection

- **Input validation** on all user inputs using Zod schemas
- **SQL injection prevention** through Prisma ORM
- **File upload restrictions** with type and size validation
- **Secure file serving** through controlled API endpoints
- **Environment variable protection** for sensitive data

Network Security

- **HTTPS enforcement** in production environments
- **CORS configuration** for cross-origin requests
- **Rate limiting** capabilities (configurable)
- **Secure headers** implementation

🚨 Reporting a Vulnerability

We take security vulnerabilities seriously. If you discover a security issue, please follow responsible disclosure:

How to Report

1. **DO NOT** create a public GitHub issue for security vulnerabilities
2. **Email us directly** at: [security@yourdomain.com] (replace with your email)
3. **Include detailed information**:
   - Description of the vulnerability

- Steps to reproduce
- Potential impact
- Suggested fix (if you have one)

What to Expect

- **Acknowledgment**: We'll respond within 48 hours
- **Assessment**: Initial assessment within 5 business days
- **Updates**: Regular updates on investigation progress
- **Resolution**: Fix timeline depends on severity
- **Disclosure**: Coordinated disclosure after fix is deployed

Response Timeline

Severity	Response Time	Fix Timeline
Critical	24 hours	7 days
High	48 hours	14 days
Medium	5 days	30 days
Low	7 days	60 days

🔧 Security Best Practices

For Developers

Input Validation

```
// Always validate user input
import { z } from 'zod'

const sessionSchema = z.object({
  title: z.string().min(1).max(100),
  description: z.string().max(500).optional(),
  isPublic: z.boolean()
})

// Use the schema to validate
const validatedData = sessionSchema.parse(userInput)
```

File Upload Security

```
// Validate file types and sizes
const allowedTypes = ['.jpg', '.jpeg', '.png', '.gif', '.pdf']
const maxSize = 10 * 1024 * 1024 // 10MB

if (!allowedTypes.includes(fileExtension)) {
  throw new Error('Invalid file type')
}

if (fileSize > maxSize) {
  throw new Error('File too large')
}
```

Database Security

```
// Use Prisma parameterized queries (automatic protection)
const user = await prisma.user.findUnique({
  where: { id: userId }, // Safe from SQL injection
  select: { id: true, name: true } // Only select needed fields
})
```

For Administrators

Environment Security

```
# Use strong secrets
NEXTAUTH_SECRET=$(openssl rand -base64 32)

# Use environment-specific URLs
NEXTAUTH_URL="https://yourdomain.com"  # Production
# NEXTAUTH_URL="http://localhost:3000"  # Development only

# Secure database connections
DATABASE_URL="postgresql://user:password@host:5432/db?sslmode=require"
```

File System Security

```
# Set proper permissions for uploads
chmod 755 uploads/
chown www-data:www-data uploads/

# Regular cleanup of temporary files
find uploads/ -type f -mtime +30 -delete
```

For Deployments

HTTPS Configuration

- **Always use HTTPS** in production
- **Configure secure headers**
- **Use proper SSL certificates**
- **Enable HSTS** (HTTP Strict Transport Security)

Database Security

- **Use SSL connections** for database

- **Regular database backups**
- **Principle of least privilege** for database users
- **Network isolation** for database servers

Server Security

- **Keep systems updated**
- **Use firewalls** to restrict access
- **Monitor logs** for suspicious activity
- **Regular security audits**

🔍 Security Checklist

Pre-Deployment

- [ ] All dependencies updated to latest secure versions
- [ ] Environment variables properly configured
- [ ] HTTPS configured and tested
- [ ] Database connections use SSL
- [ ] File upload restrictions in place
- [ ] Authentication flows tested
- [ ] Input validation on all forms
- [ ] Error messages don’t leak sensitive information

Post-Deployment

- [ ] Security headers configured
- [ ] Monitoring and logging in place
- [ ] Regular backup schedule established
- [ ] Access controls reviewed
- [ ] Incident response plan documented

Regular Maintenance

- [ ] Monthly dependency updates
- [ ] Quarterly security reviews
- [ ] Regular penetration testing
- [ ] Log review and analysis
- [ ] Backup restoration testing

🚨 Known Security Considerations

File Uploads

- **Risk**: Malicious file uploads
- **Mitigation**: File type validation, size limits, virus scanning
- **Monitoring**: Track unusual upload patterns

Session Management

- **Risk**: Session hijacking
- **Mitigation**: Secure session tokens, HTTPS, session timeout

- **Monitoring**: Monitor for unusual session activity

Database Access

- **Risk**: Unauthorized data access
- **Mitigation**: Role-based access, encrypted connections
- **Monitoring**: Database query logging

📚 Security Resources

Dependencies

We use these security-focused packages:
- **NextAuth.js**: Authentication and session management
- **Prisma**: ORM with built-in SQL injection protection
- **Zod**: Runtime type validation
- **bcryptjs**: Secure password hashing

External Resources

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

🏆 Security Credits

We appreciate security researchers and developers who help improve our security:

- Report security issues responsibly
- Contribute security improvements
- Review and suggest security enhancements

Recognition

Contributors who report valid security issues will be:
- Credited in release notes (if desired)
- Listed in our security acknowledgments
- Invited to test fixes before public release

Security is everyone's responsibility. Thank you for helping keep our users safe! 🛡️