Security Best Practices for Environment Variables

Purpose

This guide outlines security best practices for managing environment variables in the Mindful Champion application.

🔒 Critical Security Principles

1. Never Commit Secrets to Version Control

❌ **DON'T**:

```
# Committing .env file
git add .env
git commit -m "Add environment variables"
```

✅ **DO**:

```
# Ensure .env is in .gitignore
echo ".env" >> .gitignore
echo ".env.local" >> .gitignore
echo ".env.*.local" >> .gitignore

# Commit only the example file
git add .env.example
git commit -m "Add environment variables template"
```

Why: Once committed, secrets remain in git history even if deleted later.

2. Use Different Values for Different Environments

✅ **REQUIRED**: Separate values for:
- **Development** (.env.development)
- **Staging** (.env.staging)
- **Production** (.env.production)

Example:

```
# Development
NEXTAUTH_URL=http://localhost:3000
STRIPE_SECRET_KEY=sk_test_...

# Production
NEXTAUTH_URL=https://mindfulchampion.com
STRIPE_SECRET_KEY=sk_live_...
```

3. Rotate Secrets Regularly

✅ **Rotation Schedule**:

Secret	Frequency	Method
NEXTAUTH_SECRET	Every 90 days	`openssl rand -base64 32`
CRON_SECRET	Every 90 days	`openssl rand -base64 32`
GMAIL_APP_PASSWORD	Every 90 days	Generate new in Google Account
API Keys	As needed	Regenerate in service dashboard
Database passwords	Every 180 days	Update in database provider

Set a calendar reminder!

4. Principle of Least Privilege

✅ **Apply restrictions**:

API Keys:
- Restrict by domain (Stripe, Google, etc.)
- Restrict by IP address when possible
- Set usage limits
- Enable monitoring and alerts

Database:
- Use dedicated user per environment
- Grant only necessary permissions
- Enable SSL/TLS
- Use connection pooling

S3 Buckets:
- Use IAM roles with minimal permissions
- Enable versioning
- Enable access logging
- Block public access unless needed

🛡️ Security Checklist

Environment Files

- ☐ .env is in .gitignore
- ☐ .env.local is in .gitignore
- ☐ .env.*.local is in .gitignore
- ☐ No .env files committed to git
- ☐ .env.example has no real values

- ☐ Different values for dev/staging/prod

Secrets Management

- ☐ NEXTAUTH_SECRET is 32+ characters
- ☐ CRON_SECRET is 32+ characters
- ☐ All secrets are random (not guessable)
- ☐ Secrets rotation schedule set
- ☐ Old secrets revoked after rotation

Access Control

- ☐ API keys have domain restrictions
- ☐ Database uses SSL/TLS
- ☐ S3 bucket has proper IAM policies
- ☐ Stripe keys restricted by domain
- ☐ Gmail App Password is dedicated

Monitoring

- ☐ Failed authentication attempts logged
- ☐ Unusual API usage monitored
- ☐ Security alerts configured
- ☐ Regular security audits scheduled

🔐 Secure Storage Options

Local Development

Option 1: .env file (Default)

```
# Store in project root
# Ensure it's in .gitignore
.env
```

Option 2: System Environment

```
# Add to ~/.bashrc or ~/.zshrc
export DATABASE_URL="postgresql://..."
export NEXTAUTH_SECRET="..."
```

Option 3: Password Manager
- Use 1Password, LastPass, or Bitwarden
- Store entire .env file as secure note
- Share with team securely

Production Deployment

Vercel (Recommended):

```
# Set via CLI
vercel env add DATABASE_URL

# Or via Dashboard
# Project Settings → Environment Variables
```

Railway:

```
# Set in Dashboard
# Variables → Add Variable
```

Netlify:

```
# Set via CLI
netlify env:set DATABASE_URL "value"

# Or via Dashboard
# Site settings → Environment variables
```

Docker:

```
# Use secrets management
docker secret create db_password password.txt

# Or env file (not committed)
docker run --env-file .env.production app
```

🚨 What to Do If Secrets Are Compromised

Immediate Actions

1. **Revoke compromised secrets immediately**
   - Generate new NEXTAUTH_SECRET
   - Generate new CRON_SECRET
   - Revoke API keys
   - Change database password

2. **Update all environments**
   - Development
   - Staging
   - Production

3. **Deploy new version**
   ```bash
   # Update environment variables
   vercel env add NEXTAUTH_SECRET
   ```

# Redeploy
vercel –prod
```

1. **Monitor for suspicious activity**
   - Check logs for unusual access
   - Monitor API usage
   - Review database connections
   - Check email sending patterns

2. **Notify affected parties**
   - If user data affected, notify users
   - Report to relevant authorities if required

### Investigation Checklist

- ☐ Review git history for commits
- ☐ Check CI/CD logs
- ☐ Review deployment logs
- ☐ Check error monitoring services
- ☐ Review access logs
- ☐ Audit team member access

---

## 🔍 Monitoring & Alerts

### Set Up Monitoring

#### 1. Failed Authentication Attempts

```
// Log failed cron authentication
if (token !== process.env.CRON_SECRET) {
  console.error('[SECURITY] Unauthorized cron attempt', {
    timestamp: new Date().toISOString(),
    ip: request.headers.get('x-forwarded-for'),
    path: request.url,
  });
}
```

2. API Rate Limiting

```
import rateLimit from 'express-rate-limit';

const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100, // limit each IP to 100 requests per windowMs
  message: 'Too many requests from this IP',
});
```

3. Database Connection Monitoring
```

```
// Monitor connection pool
if (connectionPool.size > threshold) {
  console.warn('[SECURITY] Unusual database activity', {
    poolSize: connectionPool.size,
    timestamp: new Date().toISOString(),
  });
}
```

4. Email Sending Patterns

```
// Track daily email volume
if (dailyEmailCount > 450) { // Gmail limit is 500
  console.warn('[SECURITY] Approaching Gmail daily limit', {
    count: dailyEmailCount,
    limit: 500,
  });
}
```

Alert Configuration

Vercel:

```
# Set up alerts in Vercel Dashboard
# Project → Settings → Alerts
# Configure for:
# - Failed deployments
# - High error rates
# - Unusual traffic patterns
```

Sentry (Error Monitoring):

```
import * as Sentry from '@sentry/nextjs';

Sentry.init({
  dsn: process.env.SENTRY_DSN,
  tracesSampleRate: 1.0,
});
```

📝 Audit Trail

Keep Records

1. Secret Rotation Log

```
# SECRETS_ROTATION_LOG.md

## NEXTAUTH_SECRET
- 2025-12-03: Rotated for initial production deployment
- Next rotation: 2026-03-03

## CRON_SECRET
- 2025-12-03: Set to provided value
- Next rotation: 2026-03-03

## GMAIL_APP_PASSWORD
- 2025-12-03: Generated for welcomefrommc@mindfulchampion.com
- Next rotation: 2026-03-03
```

2. Environment Changes Log

```
# ENV_CHANGES_LOG.md

## 2025-12-03
- Added CRON_SECRET for notification system
- Updated NEXT_PUBLIC_APP_URL for production
- Added NOTIFICATION_EMAIL
- Added SUPPORT_EMAIL
```

3. Access Log

```
# ACCESS_LOG.md

## Team Access to Secrets
- Dean: Full access to production .env
- Developer 1: Development .env only
- Developer 2: Development .env only
```

🧑‍💻 Team Security

Onboarding New Team Members

1. **Provide access to development secrets only**
   `bash`
   `# Share .env.development via secure channel`
   `# Do NOT share production secrets initially`

2. **Use 1Password or similar for sharing**
   - Create shared vault
   - Add team members
   - Share relevant secrets

3. **Document access level**
   - Update ACCESS_LOG.md
   - Set appropriate permissions

Offboarding Team Members

1. **Revoke access immediately**
   - Remove from 1Password vault
   - Remove from deployment platforms
   - Remove from database access

2. **Rotate all secrets they had access to**
   - Generate new NEXTAUTH_SECRET
   - Generate new CRON_SECRET
   - Rotate API keys

3. **Update team documentation**
   - Remove from ACCESS_LOG.md
   - Update team roles

🔧 Developer Best Practices

Code Reviews

✅ **Check for**:
- No hardcoded secrets
- Proper use of `process.env`
- No secrets in logs
- No secrets in error messages
- No secrets in URLs

❌ **Red Flags**:

```
// DON'T: Hardcoded secret
const apiKey = 'sk_live_abc123';

// DON'T: Secret in log
console.log('API Key:', process.env.STRIPE_SECRET_KEY);

// DON'T: Secret in error message
throw new Error(`Failed with key: ${process.env.API_KEY}`);

// DON'T: Secret in URL
fetch(`https://api.com?key=${process.env.API_KEY}`);
```

✅ **Correct Usage**:

```javascript
// DO: Use environment variable
const apiKey = process.env.STRIPE_SECRET_KEY;

// DO: Log without exposing secret
console.log('API Key configured:', !!process.env.STRIPE_SECRET_KEY);

// DO: Generic error message
throw new Error('API authentication failed');

// DO: Secret in header
fetch('https://api.com', {
  headers: { 'Authorization': `Bearer ${process.env.API_KEY}` }
});
```

Testing

Use test values for development:

```
# .env.test
STRIPE_SECRET_KEY=sk_test_abc123
STRIPE_PUBLISHABLE_KEY=pk_test_abc123
GMAIL_USER=test@example.com
GMAIL_APP_PASSWORD=testpassword1234
```

Mock external services:

```javascript
// jest.config.js
module.exports = {
  setupFiles: ['<rootDir>/jest.setup.js'],
};

// jest.setup.js
process.env.DATABASE_URL = 'postgresql://test:test@localhost:5432/test';
process.env.NEXTAUTH_SECRET = 'test-secret-32-characters-long';
```

📊 Security Metrics

Track These Metrics

1. **Time since last rotation**
   - NEXTAUTH_SECRET: ___ days
   - CRON_SECRET: ___ days
   - API Keys: ___ days

2. **Failed authentication attempts**
   - Cron endpoints: ___ attempts
   - API endpoints: ___ attempts

3. **Unusual activity**
   - Database connections: ___ concurrent
   - API calls: ___ per hour
   - Email sending: ___ per day

Regular Security Audits

Monthly:
- ☐ Review access logs
- ☐ Check for exposed secrets (git history)
- ☐ Verify API key restrictions
- ☐ Review team access

Quarterly:
- ☐ Rotate all secrets
- ☐ Update dependencies
- ☐ Security vulnerability scan
- ☐ Penetration testing

Annually:
- ☐ Full security audit
- ☐ Compliance review
- ☐ Disaster recovery drill
- ☐ Update security policies

🚀 Quick Security Checklist

Before Every Deployment:
- ☐ No secrets in code
- ☐ All secrets set in platform
- ☐ .env not committed
- ☐ API keys restricted
- ☐ Monitoring enabled

After Every Deployment:
- ☐ Verify app works
- ☐ Check error logs
- ☐ Test critical features
- ☐ Monitor for alerts

During Security Incident:
- ☐ Revoke compromised secrets
- ☐ Update all environments
- ☐ Redeploy application
- ☐ Monitor for abuse
- ☐ Notify affected parties

📚 Additional Resources

Tools

- **git-secrets**: Prevent committing secrets
  ```
  bash
    brew install git-secrets
  ```

```
  git secrets --install
  git secrets --register-aws
```

- **truffleHog**: Find secrets in git history
  ```
  bash
    pip install truffleHog
    truffleHog --regex --entropy=False .
  ```

- **dotenv-vault**: Encrypted .env files
  ```
  bash
    npm install dotenv-vault
    npx dotenv-vault new
  ```

Documentation

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- NIST Cybersecurity Framework (https://www.nist.gov/cyberframework)
- Vercel Security (https://vercel.com/docs/security)
- Next.js Security (https://nextjs.org/docs/pages/building-your-application/deploying#security)

Last Updated: December 3, 2025
Next Security Audit: March 3, 2026
Status: Production Ready