🔒 Política de Seguridad - EscalaFin MVP

Versiones Soportadas

Actualmente mantenemos las siguientes versiones con actualizaciones de seguridad:

Versión	Soportada
2.1.x	✅ Sí
2.0.x	✅ Sí
1.2.x	⚠️ Solo críticos
< 1.2	❌ No

🚨 Reportar Vulnerabilidades

Proceso de Reporte

Si descubres una vulnerabilidad de seguridad, por favor:

1. **NO** abras un issue público
2. Envía un email a: `security@escalafin.com`
3. Incluye la información detallada que se solicita abajo

Información Requerida

```
- Descripción de la vulnerabilidad
- Pasos detallados para reproducir
- Versiones afectadas
- Impacto potencial
- Posibles soluciones (si las conoces)
```

Tiempo de Respuesta

- **Confirmación inicial**: Dentro de 48 horas
- **Evaluación completa**: Dentro de 5 días hábiles
- **Resolución**: Dependiendo de la severidad
- Crítica: 24-48 horas
- Alta: 1 semana
- Media: 2-3 semanas
- Baja: 1-2 meses

🛡️ Medidas de Seguridad Implementadas

Autenticación y Autorización

```
// NextAuth.js con JWT seguros
export const authOptions: NextAuthOptions = {
  providers: [
    CredentialsProvider({
      // Validación segura de credenciales
      async authorize(credentials) {
        // Hash con bcrypt
        const isValid = await bcrypt.compare(password, hashedPassword);
        return isValid ? user : null;
      }
    })
  ],
  session: {
    strategy: "jwt",
    maxAge: 30 * 60, // 30 minutos
  },
  jwt: {
    secret: process.env.NEXTAUTH_SECRET,
  }
};
```

Validación de Entrada

```
// Zod schemas para validación
export const userSchema = z.object({
  email: z.string().email("Email inválido"),
  password: z.string()
    .min(8, "Mínimo 8 caracteres")
    .regex(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)/, "Debe contener mayúscula, minúscula y
número"),
  name: z.string()
    .min(2, "Mínimo 2 caracteres")
    .max(100, "Máximo 100 caracteres")
    .regex(/^[a-zA-ZáéíóúÁÉÍÓÚñÑ\s]+$/, "Solo letras y espacios"),
});
```

Sanitización de Datos

```
import DOMPurify from 'dompurify';

// Sanitizar HTML
const sanitizeHtml = (dirty: string): string => {
  return DOMPurify.sanitize(dirty);
};

// Escape SQL injection (usando Prisma ORM)
const getUserById = async (id: string) => {
  return await db.user.findUnique({
    where: { id }, // Prisma maneja la sanitización
  });
};
```

Rate Limiting

```
// middleware.ts
import { NextResponse } from 'next/server';
import type { NextRequest } from 'next/server';

const rateLimitMap = new Map();

export function middleware(request: NextRequest) {
  const ip = request.ip || '127.0.0.1';
  const now = Date.now();
  const windowMs = 15 * 60 * 1000; // 15 minutos
  const maxRequests = 100;

  if (!rateLimitMap.has(ip)) {
    rateLimitMap.set(ip, { count: 1, resetTime: now + windowMs });
    return NextResponse.next();
  }

  const userData = rateLimitMap.get(ip);

  if (now > userData.resetTime) {
    userData.count = 1;
    userData.resetTime = now + windowMs;
  } else {
    userData.count++;
  }

  if (userData.count > maxRequests) {
    return new NextResponse('Too Many Requests', { status: 429 });
  }

  return NextResponse.next();
}
```

Headers de Seguridad

```
// next.config.js
const securityHeaders = [
  {
    key: 'X-DNS-Prefetch-Control',
    value: 'on'
  },
  {
    key: 'X-XSS-Protection',
    value: '1; mode=block'
  },
  {
    key: 'X-Frame-Options',
    value: 'SAMEORIGIN'
  },
  {
    key: 'X-Content-Type-Options',
    value: 'nosniff'
  },
  {
    key: 'Referrer-Policy',
    value: 'origin-when-cross-origin'
  },
  {
    key: 'Content-Security-Policy',
    value: "default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline';
style-src 'self' 'unsafe-inline';"
  }
];

module.exports = {
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: securityHeaders,
      },
    ];
  },
};
```

🔐 Configuración Segura

Variables de Entorno

```
# ❌ MAL - Nunca hacer esto
API_KEY=sk-live-abc123  # En código fuente

# ✅ BIEN - Usar variables de entorno
API_KEY=${SECURE_API_KEY}
```

Secrets Management

```
// Usar variables de entorno para secrets
const config = {
  database: {
    url: process.env.DATABASE_URL!,
  },
  auth: {
    secret: process.env.NEXTAUTH_SECRET!,
  },
  openpay: {
    merchantId: process.env.OPENPAY_MERCHANT_ID!,
    privateKey: process.env.OPENPAY_PRIVATE_KEY!,
  }
};

// Validar que existan las variables críticas
const requiredEnvVars = [
  'DATABASE_URL',
  'NEXTAUTH_SECRET',
  'OPENPAY_MERCHANT_ID'
];

requiredEnvVars.forEach(envVar => {
  if (!process.env[envVar]) {
    throw new Error(`Missing required environment variable: ${envVar}`);
  }
});
```

🔍 Auditoría y Monitoreo

Logging de Seguridad

```
// lib/security-logger.ts
export const securityLogger = {
  loginAttempt: (email: string, success: boolean, ip: string) => {
    console.log(JSON.stringify({
      event: 'LOGIN_ATTEMPT',
      email: email.replace(/(.{2})(.*)(@.*)/, '$1***$3'), // Enmascarar email
      success,
      ip,
      timestamp: new Date().toISOString()
    }));
  },

  suspiciousActivity: (userId: string, activity: string, ip: string) => {
    console.warn(JSON.stringify({
      event: 'SUSPICIOUS_ACTIVITY',
      userId,
      activity,
      ip,
      timestamp: new Date().toISOString()
    }));
  }
};
```

Monitoreo de Errores

```
// lib/error-monitoring.ts
export const monitorError = (error: Error, context: Record<string, any>) => {
  // En producción, enviar a servicio de monitoreo
  if (process.env.NODE_ENV === 'production') {
    // Sentry, Bugsnag, etc.
    console.error({
      message: error.message,
      stack: error.stack,
      context,
      timestamp: new Date().toISOString()
    });
  }
};
```

🛠️ Herramientas de Seguridad

Dependencias

```
{
  "devDependencies": {
    "audit": "^2.0.0",
    "eslint-plugin-security": "^1.7.0",
    "helmet": "^7.0.0"
  }
}
```

Scripts de Seguridad

```
{
  "scripts": {
    "security:audit": "yarn audit --audit-level moderate",
    "security:check": "eslint . --ext .ts,.tsx --config .eslintrc-security.js",
    "security:deps": "yarn audit --json | audit-ci --config audit-ci.json"
  }
}
```

ESLint Security Plugin

```
// .eslintrc-security.js
module.exports = {
  plugins: ['security'],
  extends: ['plugin:security/recommended'],
  rules: {
    'security/detect-object-injection': 'error',
    'security/detect-non-literal-regexp': 'error',
    'security/detect-unsafe-regex': 'error',
    'security/detect-buffer-noassert': 'error',
    'security/detect-child-process': 'error',
    'security/detect-disable-mustache-escape': 'error',
    'security/detect-eval-with-expression': 'error',
    'security/detect-no-csrf-before-method-override': 'error',
    'security/detect-non-literal-fs-filename': 'error',
    'security/detect-non-literal-require': 'error',
    'security/detect-possible-timing-attacks': 'error',
    'security/detect-pseudoRandomBytes': 'error'
  }
};
```

🧪 Testing de Seguridad

Tests de Autenticación

```
// __tests__/security/auth.test.ts
describe('Authentication Security', () => {
  test('should reject weak passwords', async () => {
    const weakPassword = '123';
    const result = await validatePassword(weakPassword);
    expect(result.isValid).toBe(false);
  });

  test('should hash passwords securely', async () => {
    const password = 'SecurePassword123!';
    const hashed = await hashPassword(password);
    expect(hashed).not.toBe(password);
    expect(await comparePassword(password, hashed)).toBe(true);
  });

  test('should prevent SQL injection', async () => {
    const maliciousInput = "'; DROP TABLE users; --";
    const result = await getUserById(maliciousInput);
    expect(result).toBe(null);
  });
});
```

Tests de Autorización

```
describe('Authorization Security', () => {
  test('should deny access to admin routes for non-admin users', async () => {
    const clientUser = { role: 'CLIENTE' };
    const hasAccess = await checkAdminAccess(clientUser);
    expect(hasAccess).toBe(false);
  });

  test('should allow access only to own data', async () => {
    const user1 = { id: '1', role: 'CLIENTE' };
    const user2Data = { userId: '2', name: 'User 2' };

    const canAccess = await canAccessUserData(user1, user2Data);
    expect(canAccess).toBe(false);
  });
});
```

🔄 Actualizaciones de Seguridad

Proceso de Actualización

1. **Monitoreo continuo** de vulnerabilidades
2. **Evaluación de impacto** de cada vulnerabilidad
3. **Priorización** basada en severidad
4. **Testing exhaustivo** antes de deploy
5. **Comunicación** a usuarios si es necesario

Dependencias

```
# Auditoría regular
yarn audit

# Actualización de dependencias
yarn upgrade-interactive --latest

# Verificación de vulnerabilidades conocidas
yarn audit --level moderate
```

📋 Checklist de Seguridad

Desarrollo

- [ ] Validar todas las entradas de usuario
- [ ] Sanitizar salidas HTML
- [ ] Usar prepared statements (ORM)
- [ ] Implementar rate limiting
- [ ] Logs de seguridad apropiados
- [ ] Headers de seguridad configurados
- [ ] Autenticación robusta
- [ ] Autorización granular

Despliegue

- [ ] HTTPS configurado
- [ ] Variables de entorno seguras
- [ ] Base de datos con acceso restringido
- [ ] Firewall configurado
- [ ] Backups encriptados
- [ ] Monitoreo activo
- [ ] Certificados SSL válidos

Mantenimiento

- [ ] Actualizaciones regulares de dependencias
- [ ] Auditorías de seguridad periódicas
- [ ] Review de logs de seguridad
- [ ] Tests de penetración ocasionales
- [ ] Capacitación del equipo
- [ ] Documentación actualizada

🎓 Recursos de Seguridad

Documentación

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)
- Next.js Security (https://nextjs.org/docs/app/building-your-application/configuring/security)
- Node.js Security Checklist (https://blog.risingstack.com/node-js-security-checklist/)

Herramientas

- Snyk (https://snyk.io/) - Análisis de vulnerabilidades
- ESLint Security Plugin (https://github.com/nodesecurity/eslint-plugin-security)
- Helmet.js (https://helmetjs.github.io/) - Headers de seguridad

---

La seguridad es responsabilidad de todos. Si ves algo, di algo. 🔒

Para reportes de seguridad urgentes: `security@escalafin.com`