Medidas de Segurança e Proteção de Dados para o E-commerce

Visão Geral

Este documento detalha as medidas de segurança e proteção de dados que serão implementadas no e-commerce, seguindo as melhores práticas do mercado e em conformidade com a Lei Geral de Proteção de Dados (LGPD) do Brasil. A segurança e a proteção de dados são prioridades máximas neste projeto, conforme solicitado pelo cliente.

Princípios Fundamentais de Segurança

1. Defesa em Profundidade

Implementaremos múltiplas camadas de segurança em todo o sistema, de modo que se uma camada for comprometida, outras camadas ainda protegerão os dados e a funcionalidade do sistema.

2. Princípio do Menor Privilégio

Cada componente do sistema terá acesso apenas aos recursos e dados necessários para executar suas funções específicas, minimizando o impacto potencial de uma violação.

3. Segurança por Design

A segurança será considerada em todas as fases do desenvolvimento, desde o planejamento inicial até a implementação e manutenção contínua.

4. Privacidade por Design

A proteção de dados pessoais será incorporada ao design do sistema desde o início, garantindo que a privacidade seja considerada em todas as decisões de design.

Segurança da Infraestrutura

1. Hospedagem Segura

- **HTTPS Obrigatório**: Todo o site utilizará HTTPS com certificados SSL/TLS válidos para criptografar todas as comunicações.
- **Configuração Segura de Servidor**: Implementação de cabeçalhos de segurança HTTP como:
- Content-Security-Policy (CSP)
- Strict-Transport-Security (HSTS)
- X-Content-Type-Options
- X-Frame-Options
- X-XSS-Protection

2. Firewall e Proteção de Rede

- **Web Application Firewall (WAF)**: Implementação de WAF para filtrar e monitorar o tráfego HTTP.
- **Proteção DDoS**: Medidas para mitigar ataques de negação de serviço distribuído.
- **Monitoramento de Rede**: Sistemas para detectar comportamentos anômalos e possíveis intrusões.

3. Atualizações e Patches

- **Atualizações Regulares**: Manutenção de todos os componentes do sistema com as versões mais recentes e seguras.
- **Gerenciamento de Patches**: Processo para aplicar patches de segurança de forma rápida e eficiente.
- **Monitoramento de Vulnerabilidades**: Acompanhamento contínuo de novas vulnerabilidades que possam afetar o sistema.

Segurança de Aplicação

1. Autenticação e Autorização

Autenticação Robusta

- **Senhas Fortes**: Exigência de senhas complexas com pelo menos 8 caracteres, incluindo letras maiúsculas, minúsculas, números e caracteres especiais.
- **Hashing Seguro de Senhas**: Utilização de algoritmos modernos como Argon2 ou bcrypt com salt único para cada usuário.

- **Proteção Contra Força Bruta**: Implementação de limitação de tentativas de login (rate limiting) e bloqueio temporário de contas após múltiplas tentativas falhas.
- **Autenticação de Dois Fatores (2FA)**: Opção para usuários habilitarem 2FA via aplicativos autenticadores ou SMS.
- **Gerenciamento de Sessão Seguro**: Tokens JWT com tempo de expiração curto e rotação de tokens.

```
// Exemplo de implementação de hashing seguro de senhas com bcrypt
import bcrypt from 'bcryptjs';

// Gerar hash de senha
export async function hashPassword(password) {
  const saltRounds = 12; // Custo computacional alto para dificultar ataques
  return await bcrypt.hash(password, saltRounds);
}

// Verificar senha
export async function verifyPassword(password, hashedPassword) {
  return await bcrypt.compare(password, hashedPassword);
}
```

Autorização Granular

- **Controle de Acesso Baseado em Funções (RBAC)**: Diferentes níveis de acesso para clientes, administradores e outros papéis.
- **Verificação de Permissões em Cada Requisição**: Validação de permissões em todas as operações sensíveis.
- **Princípio do Menor Privilégio**: Cada usuário terá apenas as permissões necessárias para suas funções.

```
// Exemplo de middleware para verificação de permissões
export function requirePermission(permission) {
  return async (req, res, next) => {
    const user = req.user;

    if (!user) {
      return res.status(401).json({ error: 'Autenticação necessária' });
    }

    const hasPermission = await checkUserPermission(user.id, permission);

    if (!hasPermission) {
      return res.status(403).json({ error: 'Acesso negado' });
    }

    next();
```

```
  };
}
```

2. Proteção Contra Vulnerabilidades Comuns

Injeção de SQL

- **ORM com Consultas Parametrizadas**: Uso do Prisma ORM que previne injeção de SQL por design.
- **Validação de Entrada**: Verificação rigorosa de todos os dados de entrada antes de processá-los.
- **Escape de Caracteres Especiais**: Tratamento adequado de caracteres que poderiam ser usados em ataques de injeção.

```
// Exemplo de consulta segura com Prisma (previne injeção de SQL)
const user = await prisma.user.findUnique({
  where: { email: userInput.email } // Prisma usa consultas parametrizadas automaticamente
});
```

Cross-Site Scripting (XSS)

- **Sanitização de Entrada**: Limpeza de todos os dados de entrada para remover scripts maliciosos.
- **Escape de Saída**: Codificação adequada de dados antes de renderizá-los no navegador.
- **Content Security Policy (CSP)**: Implementação de políticas CSP para restringir fontes de conteúdo.
- **HttpOnly e Secure Cookies**: Proteção de cookies contra acesso por JavaScript e transmissão apenas por HTTPS.

```
// Exemplo de sanitização de entrada com DOMPurify
import DOMPurify from 'dompurify';

function sanitizeHtml(content) {
  return DOMPurify.sanitize(content, {
    ALLOWED_TAGS: ['b', 'i', 'em', 'strong', 'p', 'br'],
    ALLOWED_ATTR: []
  });
}

// Antes de salvar conteúdo gerado pelo usuário
const safeContent = sanitizeHtml(userInput.description);
```

Cross-Site Request Forgery (CSRF)

- **Tokens CSRF**: Geração e validação de tokens anti-CSRF para todas as requisições que modificam dados.
- **Same-Site Cookies**: Configuração de cookies como SameSite=Strict ou SameSite=Lax.
- **Verificação de Origem**: Validação do cabeçalho Origin ou Referer em requisições sensíveis.

```
 // Exemplo de geração e validação de token CSRF
import { randomBytes, timingSafeEqual } from 'crypto';

// Gerar token CSRF
export function generateCsrfToken() {
  return randomBytes(32).toString('hex');
}

// Validar token CSRF
export function validateCsrfToken(token, storedToken) {
  if (!token || !storedToken) {
    return false;
  }

  try {
    return timingSafeEqual(
      Buffer.from(token, 'hex'),
      Buffer.from(storedToken, 'hex')
    );
  } catch (error) {
    return false;
  }
}
```

Broken Authentication

- **Gerenciamento Seguro de Credenciais**: Armazenamento seguro de credenciais com hashing e salt.
- **Políticas de Senha Fortes**: Exigência de senhas complexas e verificação contra senhas comuns/vazadas.
- **Expiração de Sessão**: Tempo limite para sessões inativas e opção "lembrar-me" segura.
- **Renovação Segura de Credenciais**: Processo seguro para redefinição de senha e recuperação de conta.

```
 // Exemplo de validação de força de senha
function isStrongPassword(password) {
```

```
  // Pelo menos 8 caracteres
  if (password.length < 8) return false;

  // Pelo menos uma letra maiúscula
  if (!/[A-Z]/.test(password)) return false;

  // Pelo menos uma letra minúscula
  if (!/[a-z]/.test(password)) return false;

  // Pelo menos um número
  if (!/[0-9]/.test(password)) return false;

  // Pelo menos um caractere especial
  if (!/[^A-Za-z0-9]/.test(password)) return false;

  // Verificar se não é uma senha comum (implementação simplificada)
  const commonPasswords = ['Password123!', 'Admin123!', '12345678Aa!'];
  if (commonPasswords.includes(password)) return false;

  return true;
}
```

Server-Side Request Forgery (SSRF)

- **Validação de URLs**: Verificação rigorosa de URLs fornecidos pelo usuário.
- **Listas Brancas**: Restrição de domínios e IPs permitidos para requisições do servidor.
- **Firewall de Saída**: Limitação de conexões de saída apenas para serviços necessários.

```
// Exemplo de validação de URL para prevenir SSRF
function isValidUrl(url) {
  try {
    const parsedUrl = new URL(url);

    // Lista branca de domínios permitidos
    const allowedDomains = ['api.mercadopago.com', 'api.correios.com.br'];

    // Verificar se o domínio está na lista branca
    if (!allowedDomains.includes(parsedUrl.hostname)) {
      return false;
    }

    // Verificar se não é um IP interno
    if (/^(127\.|10\.|172\.(1[6-9]|2[0-9]|3[0-1])\.|192\.
168\.)/.test(parsedUrl.hostname)) {
      return false;
    }
```

```
    return true;
  } catch (error) {
    return false;
  }
}
```

Insecure Deserialization

- **Validação de Dados Serializados**: Verificação rigorosa de dados antes da desserialização.
- **Uso de Formatos Seguros**: Preferência por formatos como JSON em vez de serialização nativa.
- **Assinatura Digital**: Assinatura de dados serializados para garantir integridade.

```
// Exemplo de desserialização segura com validação de esquema
import { z } from 'zod';

// Definir esquema de validação
const userSchema = z.object({
  id: z.number().int().positive(),
  name: z.string().min(1).max(100),
  email: z.string().email(),
  role: z.enum(['customer', 'admin'])
});

// Desserializar dados com validação
function deserializeUser(data) {
  try {
    // Analisar JSON
    const parsed = JSON.parse(data);

    // Validar contra esquema
    const result = userSchema.safeParse(parsed);

    if (!result.success) {
      throw new Error('Dados inválidos');
    }

    return result.data;
  } catch (error) {
    // Lidar com erro de desserialização
    console.error('Erro na desserialização:', error);
    return null;
  }
}
```

XML External Entities (XXE)

- **Desativação de Entidades Externas**: Configuração de parsers XML para desativar entidades externas.
- **Uso de Formatos Alternativos**: Preferência por JSON em vez de XML quando possível.
- **Validação de Esquema**: Uso de validação de esquema XML para verificar a estrutura dos documentos.

```
// Exemplo de configuração segura de parser XML
import { DOMParser } from 'xmldom';

function parseXmlSafely(xmlString) {
 // Configurar parser para desativar entidades externas
 const parser = new DOMParser({
  errorHandler: {
   warning: () => {},
   error: () => {},
   fatalError: (e) => { throw e; }
  }
 });

 // Opções de segurança
 const options = {
  resolveExternalEntities: false,
  validateOnParse: true
 };

 return parser.parseFromString(xmlString, 'text/xml', options);
}
```

3. Proteção de Dados Sensíveis

Criptografia em Trânsito

- **HTTPS em Todo o Site**: Uso de HTTPS para todas as comunicações.
- **Certificados SSL/TLS Atualizados**: Uso de certificados válidos e algoritmos de criptografia modernos.
- **HTTP Strict Transport Security (HSTS)**: Configuração para forçar conexões HTTPS.

Criptografia em Repouso

- **Criptografia de Dados Sensíveis**: Armazenamento criptografado de informações como dados de pagamento.

- **Chaves de Criptografia Seguras**: Gerenciamento seguro de chaves de criptografia com rotação regular.
- **Tokenização**: Uso de tokens em vez de dados reais para informações sensíveis.

```
// Exemplo de criptografia de dados sensíveis
import { createCipheriv, createDecipheriv, randomBytes } from 'crypto';

// Criptografar dados
export function encrypt(text, masterKey) {
  // Gerar IV (Vetor de Inicialização) único para cada operação
  const iv = randomBytes(16);

  // Criar cipher com algoritmo AES-256-GCM
  const cipher = createCipheriv('aes-256-gcm', masterKey, iv);

  // Criptografar dados
  let encrypted = cipher.update(text, 'utf8', 'hex');
  encrypted += cipher.final('hex');

  // Obter tag de autenticação
  const authTag = cipher.getAuthTag();

  // Retornar IV, dados criptografados e tag de autenticação
  return {
    iv: iv.toString('hex'),
    encrypted,
    authTag: authTag.toString('hex')
  };
}

// Descriptografar dados
export function decrypt(encryptedData, masterKey) {
  // Converter IV e tag de autenticação de hex para Buffer
  const iv = Buffer.from(encryptedData.iv, 'hex');
  const authTag = Buffer.from(encryptedData.authTag, 'hex');

  // Criar decipher
  const decipher = createDecipheriv('aes-256-gcm', masterKey, iv);

  // Definir tag de autenticação
  decipher.setAuthTag(authTag);

  // Descriptografar dados
  let decrypted = decipher.update(encryptedData.encrypted, 'hex', 'utf8');
  decrypted += decipher.final('utf8');

  return decrypted;
}
```

Mascaramento de Dados

- **Mascaramento de Informações Sensíveis**: Exibição parcial de números de cartão, CPF, etc.
- **Logs Sanitizados**: Remoção de dados sensíveis dos logs do sistema.
- **Dados de Teste Seguros**: Uso de dados fictícios para testes e desenvolvimento.

```
// Exemplo de mascaramento de dados sensíveis
function maskCreditCard(cardNumber) {
  // Manter apenas os primeiros 6 e últimos 4 dígitos
  return cardNumber.replace(/^(\d{6})\d+(\d{4})$/, '$1******$2');
}

function maskCpf(cpf) {
  // Formato: XXX.XXX.XXX-XX
  return cpf.replace(/^(\d{3})\.(\d{3})\.(\d{3})-(\d{2})$/, 'XXX.XXX.$3-$4');
}

function maskEmail(email) {
  // Formato: p***@dominio.com
  const [username, domain] = email.split('@');
  const maskedUsername = username.charAt(0) + '***';
  return `${maskedUsername}@${domain}`;
}
```

4. Monitoramento e Resposta a Incidentes

Logging e Monitoramento

- **Logs Detalhados**: Registro de todas as atividades relevantes para segurança.
- **Centralização de Logs**: Coleta e armazenamento centralizado de logs para análise.
- **Alertas em Tempo Real**: Configuração de alertas para atividades suspeitas.
- **Monitoramento de Integridade**: Verificação regular da integridade dos arquivos do sistema.

```
// Exemplo de sistema de logging seguro
import winston from 'winston';

// Configurar logger
const logger = winston.createLogger({
  level: 'info',
  format: winston.format.combine(
    winston.format.timestamp(),
    winston.format.json()
  ),
  defaultMeta: { service: 'ecommerce-api' },
  transports: [
```

```
    // Logs de console para desenvolvimento
    new winston.transports.Console({
      format: winston.format.simple(),
    }),
    // Logs de arquivo para produção
    new winston.transports.File({
      filename: 'error.log',
      level: 'error',
      maxsize: 5242880, // 5MB
      maxFiles: 5,
    }),
    new winston.transports.File({
      filename: 'combined.log',
      maxsize: 5242880, // 5MB
      maxFiles: 5,
    }),
  ],
});

// Função para sanitizar dados sensíveis nos logs
function sanitizeLogData(data) {
  const sensitiveFields = ['password', 'credit_card', 'cpf', 'token'];

  if (!data) return data;

  const sanitized = { ...data };

  for (const field of sensitiveFields) {
    if (field in sanitized) {
      sanitized[field] = '[REDACTED]';
    }
  }

  return sanitized;
}

// Função para registrar eventos de segurança
export function logSecurityEvent(eventType, data, userId = null) {
  const sanitizedData = sanitizeLogData(data);

  logger.info({
    type: 'SECURITY',
    event: eventType,
    user: userId,
    data: sanitizedData,
    timestamp: new Date().toISOString()
  });
}
```

Detecção de Intrusão

- **Sistema de Detecção de Intrusão (IDS)**: Monitoramento de atividades suspeitas.
- **Análise de Comportamento**: Detecção de padrões anômalos de uso.
- **Verificação de Vulnerabilidades**: Escaneamento regular de vulnerabilidades.

Plano de Resposta a Incidentes

- **Procedimentos Documentados**: Passos claros para responder a diferentes tipos de incidentes.
- **Equipe de Resposta**: Definição de responsabilidades e canais de comunicação.
- **Análise Pós-Incidente**: Processo para aprender com incidentes e melhorar a segurança.

```
// Exemplo de detecção de atividade suspeita
async function detectSuspiciousActivity(userId, action, data) {
 // Verificar login de localização incomum
 if (action === 'LOGIN') {
  const userIp = data.ip;
  const userAgent = data.userAgent;

  // Buscar histórico de logins do usuário
  const previousLogins = await prisma.userLogin.findMany({
   where: { userId },
   orderBy: { createdAt: 'desc' },
   take: 5
  });

  // Verificar se o IP é novo
  const isNewIp = !previousLogins.some(login => login.ip === userIp);

  if (isNewIp && previousLogins.length > 0) {
   // Registrar alerta
   await prisma.securityAlert.create({
    data: {
     userId,
     type: 'NEW_LOGIN_LOCATION',
     severity: 'MEDIUM',
     details: {
      ip: userIp,
      userAgent,
      previousIps: previousLogins.map(l => l.ip)
     }
    }
   });

   // Enviar notificação ao usuário
   await sendSecurityAlert(userId, 'new_login_location', {
    ip: userIp,
```

```
        time: new Date().toISOString()
      });
    }
  }

  // Verificar múltiplas tentativas de pagamento falhas
  if (action === 'PAYMENT_FAILED') {
    const recentFailures = await prisma.payment.count({
      where: {
        userId,
        status: 'REJECTED',
        createdAt: {
          gte: new Date(Date.now() - 24 * 60 * 60 * 1000) // Últimas 24 horas
        }
      }
    });

    if (recentFailures >= 3) {
      // Registrar alerta
      await prisma.securityAlert.create({
        data: {
          userId,
          type: 'MULTIPLE_PAYMENT_FAILURES',
          severity: 'HIGH',
          details: {
            count: recentFailures,
            latestAttempt: data
          }
        }
      });

      // Bloquear temporariamente novas tentativas de pagamento
      await prisma.user.update({
        where: { id: userId },
        data: { paymentBlocked: true, paymentBlockedUntil: new Date(Date.now() + 2
* 60 * 60 * 1000) } // 2 horas
      });
    }
  }
}
```

Proteção de Dados e Conformidade com LGPD

1. Coleta e Processamento de Dados

Minimização de Dados

- **Coleta Mínima**: Coleta apenas dos dados necessários para as finalidades declaradas.
- **Limitação de Uso**: Uso de dados pessoais apenas para os fins especificados.
- **Retenção Limitada**: Armazenamento de dados apenas pelo tempo necessário.

Consentimento Explícito

- **Termos Claros**: Termos de uso e política de privacidade claros e acessíveis.
- **Consentimento Granular**: Opções específicas para diferentes tipos de processamento de dados.
- **Revogação Fácil**: Processo simples para revogar consentimento.

```
// Exemplo de gerenciamento de consentimento
async function updateUserConsent(userId, consentType, isGranted) {
 // Verificar se o tipo de consentimento é válido
 const validConsentTypes = ['marketing_emails', 'third_party_sharing',
'analytics_cookies', 'personalization'];

 if (!validConsentTypes.includes(consentType)) {
  throw new Error('Tipo de consentimento inválido');
 }

 // Buscar consentimento existente
 const existingConsent = await prisma.userConsent.findFirst({
  where: {
   userId,
   type: consentType
  }
 });

 if (existingConsent) {
  // Atualizar consentimento existente
  await prisma.userConsent.update({
   where: { id: existingConsent.id },
   data: {
    isGranted,
    updatedAt: new Date()
   }
  });
 } else {
  // Criar novo registro de consentimento
```

```
    await prisma.userConsent.create({
      data: {
        userId,
        type: consentType,
        isGranted,
        createdAt: new Date(),
        updatedAt: new Date()
      }
    });
  }

  // Registrar alteração de consentimento
  await prisma.userConsentLog.create({
    data: {
      userId,
      type: consentType,
      isGranted,
      ipAddress: request.ip,
      userAgent: request.headers['user-agent']
    }
  });

  return { success: true };
}
```

2. Direitos dos Titulares de Dados

Acesso e Portabilidade

- **Acesso aos Dados**: Interface para usuários visualizarem seus dados pessoais.
- **Exportação de Dados**: Funcionalidade para exportar dados em formato legível por máquina.
- **Histórico de Atividades**: Registro de ações realizadas pelo usuário.

```
// Exemplo de exportação de dados do usuário
async function exportUserData(userId) {
  // Buscar dados do usuário
  const user = await prisma.user.findUnique({
    where: { id: userId },
    select: {
      id: true,
      name: true,
      email: true,
      phone: true,
      birthDate: true,
      createdAt: true,
      addresses: true,
      orders: {
        include: {
```

```
        items: true,
        payments: {
          select: {
            method: true,
            status: true,
            amount: true,
            createdAt: true
          }
        }
      }
    },
    reviews: {
      select: {
        productId: true,
        rating: true,
        comment: true,
        createdAt: true
      }
    }
  }
});

if (!user) {
  throw new Error('Usuário não encontrado');
}

// Sanitizar dados sensíveis
const sanitizedUser = {
  ...user,
  // Remover dados sensíveis que não devem ser exportados
};

// Formatar dados para exportação
const exportData = {
  personalInfo: {
    name: sanitizedUser.name,
    email: sanitizedUser.email,
    phone: sanitizedUser.phone,
    birthDate: sanitizedUser.birthDate,
    registrationDate: sanitizedUser.createdAt
  },
  addresses: sanitizedUser.addresses.map(addr => ({
    name: addr.name,
    zipCode: addr.zipCode,
    street: addr.street,
    number: addr.number,
    complement: addr.complement,
    neighborhood: addr.neighborhood,
    city: addr.city,
    state: addr.state,
    country: addr.country,
    type: addr.type
```

```
    })),
    orders: sanitizedUser.orders.map(order => ({
      code: order.code,
      date: order.createdAt,
      status: order.status,
      total: order.total,
      items: order.items.map(item => ({
        productName: item.productName,
        quantity: item.quantity,
        unitPrice: item.unitPrice,
        subtotal: item.subtotal
      })),
      payments: order.payments.map(payment => ({
        method: payment.method,
        status: payment.status,
        amount: payment.amount,
        date: payment.createdAt
      }))
    })),
    reviews: sanitizedUser.reviews.map(review => ({
      productId: review.productId,
      rating: review.rating,
      comment: review.comment,
      date: review.createdAt
    }))
  };

  // Registrar solicitação de exportação
  await prisma.dataExportRequest.create({
    data: {
      userId,
      status: 'COMPLETED',
      requestedAt: new Date(),
      completedAt: new Date()
    }
  });

  return exportData;
}
```

Correção e Exclusão

- **Edição de Dados**: Interface para usuários corrigirem seus dados pessoais.
- **Exclusão de Conta**: Processo para usuários excluírem suas contas e dados.
- **Anonimização**: Opção para anonimizar dados em vez de excluí-los completamente.

```
// Exemplo de exclusão de conta de usuário
async function deleteUserAccount(userId) {
```

```
  // Verificar se há pedidos em andamento
  const activeOrders = await prisma.order.count({
    where: {
      userId,
      status: {
        in: ['AWAITING_PAYMENT', 'PAYMENT_APPROVED', 'IN_PREPARATION',
'SHIPPED']
      }
    }
  });

  if (activeOrders > 0) {
    throw new Error('Não é possível excluir a conta com pedidos em andamento');
  }

  // Iniciar transação para garantir que todas as operações sejam concluídas ou
nenhuma
  return await prisma.$transaction(async (tx) => {
    // Anonimizar avaliações
    await tx.review.updateMany({
      where: { userId },
      data: {
        userId: null,
        name: 'Usuário anônimo',
        email: `anon_${Date.now()}@example.com`
      }
    });

    // Marcar pedidos como anônimos
    await tx.order.updateMany({
      where: { userId },
      data: { userId: null }
    });

    // Excluir dados pessoais
    await tx.address.deleteMany({ where: { userId } });
    await tx.cart.deleteMany({ where: { userId } });
    await tx.wishlist.deleteMany({ where: { userId } });
    await tx.userConsent.deleteMany({ where: { userId } });

    // Registrar exclusão
    await tx.accountDeletion.create({
      data: {
        originalUserId: userId,
        requestedAt: new Date(),
        completedAt: new Date(),
        reason: 'USER_REQUEST'
      }
    });

    // Excluir ou anonimizar o usuário
    await tx.user.delete({ where: { id: userId } });
```

```
    return { success: true };
  });
}
```

3. Medidas Técnicas e Organizacionais

Registro de Atividades de Tratamento

- **Documentação Detalhada**: Registro de todas as atividades de processamento de dados.
- **Base Legal**: Documentação da base legal para cada tipo de processamento.
- **Finalidade**: Documentação clara da finalidade de cada atividade de processamento.

Relatório de Impacto à Proteção de Dados

- **Avaliação de Riscos**: Identificação e avaliação de riscos à privacidade.
- **Medidas Mitigadoras**: Documentação de medidas para mitigar riscos identificados.
- **Revisão Regular**: Atualização periódica da avaliação de impacto.

Política de Privacidade

- **Linguagem Clara**: Política de privacidade em linguagem simples e acessível.
- **Informações Completas**: Detalhamento de todas as práticas de processamento de dados.
- **Atualizações Transparentes**: Notificação clara sobre alterações na política.

Segurança de Pagamentos

1. Conformidade com PCI DSS

- **Requisitos PCI DSS**: Implementação de todos os requisitos aplicáveis do Payment Card Industry Data Security Standard.
- **Escopo Limitado**: Minimização do escopo PCI DSS utilizando serviços de pagamento terceirizados.
- **Validação Regular**: Verificação periódica de conformidade.

2. Tokenização de Dados de Pagamento

- **Tokens em vez de Dados Reais**: Uso de tokens para representar dados de cartão de crédito.

- **Armazenamento Seguro**: Armazenamento seguro de tokens com acesso restrito.
- **Expiração de Tokens**: Implementação de tempo de vida limitado para tokens.

3. Integração Segura com Mercado Pago

- **Credenciais Seguras**: Armazenamento seguro de credenciais de API.
- **Validação de Webhooks**: Verificação da autenticidade de notificações recebidas.
- **Comunicação Criptografada**: Uso de HTTPS para todas as comunicações com a API.

```
// Exemplo de validação de webhook do Mercado Pago
import crypto from 'crypto';

function validateMercadoPagoWebhook(request) {
  const signature = request.headers['x-signature'];
  const timestamp = request.headers['x-timestamp'];
  const requestId = request.headers['x-request-id'];
  const body = request.body;

  if (!signature || !timestamp || !requestId) {
    return false;
  }

  // Verificar se o timestamp não é muito antigo (evitar replay attacks)
  const now = Math.floor(Date.now() / 1000);
  const requestTime = parseInt(timestamp, 10);

  if (now - requestTime > 300) { // 5 minutos
    return false;
  }

  // Reconstruir a string para verificação
  const data = `${requestId}${timestamp}${JSON.stringify(body)}`;

  // Calcular HMAC usando a chave secreta compartilhada
  const hmac = crypto.createHmac('sha256',
process.env.MERCADOPAGO_WEBHOOK_SECRET);
  hmac.update(data);
  const calculatedSignature = hmac.digest('hex');

  // Comparar assinaturas usando comparação de tempo constante
  return crypto.timingSafeEqual(
    Buffer.from(signature, 'hex'),
    Buffer.from(calculatedSignature, 'hex')
  );
}
```

Testes de Segurança

1. Testes Automatizados

- **Testes de Segurança Unitários**: Verificação de funções de segurança individuais.
- **Testes de Integração**: Verificação da interação segura entre componentes.
- **Testes de Regressão**: Garantia de que correções de segurança não introduzam novos problemas.

2. Análise Estática de Código

- **Ferramentas de Análise**: Uso de ferramentas como ESLint com regras de segurança.
- **Revisão de Código**: Processo de revisão por pares com foco em segurança.
- **Verificação de Dependências**: Monitoramento de vulnerabilidades em bibliotecas de terceiros.

```
// Exemplo de configuração ESLint com regras de segurança
// .eslintrc.js
module.exports = {
  extends: [
    'eslint:recommended',
    'plugin:security/recommended',
    'plugin:node/recommended'
  ],
  plugins: [
    'security',
    'node'
  ],
  rules: {
    '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-regexp': 'error',
    'security/detect-non-literal-require': 'error',
    'security/detect-object-injection': 'error',
    'security/detect-possible-timing-attacks': 'error',
    'security/detect-pseudoRandomBytes': 'error',
    'security/detect-unsafe-regex': 'error'
  }
};
```

3. Testes de Penetração

- **Testes Regulares**: Realização de testes de penetração por especialistas.
- **Escopo Abrangente**: Cobertura de todos os componentes críticos do sistema.
- **Remediação de Vulnerabilidades**: Processo para corrigir vulnerabilidades identificadas.

4. Bug Bounty

- **Programa de Recompensas**: Incentivo para pesquisadores de segurança reportarem vulnerabilidades.
- **Processo de Divulgação Responsável**: Diretrizes claras para reportar problemas de segurança.
- **Reconhecimento Público**: Agradecimento aos pesquisadores que contribuem para a segurança.

Treinamento e Conscientização

1. Treinamento da Equipe de Desenvolvimento

- **Práticas Seguras de Codificação**: Treinamento sobre como escrever código seguro.
- **Reconhecimento de Vulnerabilidades**: Capacitação para identificar problemas de segurança.
- **Atualizações Regulares**: Treinamento contínuo sobre novas ameaças e contramedidas.

2. Documentação de Segurança

- **Diretrizes de Segurança**: Documentação clara sobre práticas de segurança.
- **Procedimentos de Resposta**: Instruções detalhadas para lidar com incidentes.
- **Checklists de Segurança**: Listas de verificação para diferentes fases do desenvolvimento.

3. Cultura de Segurança

- **Priorização da Segurança**: Incorporação da segurança em todas as decisões.
- **Comunicação Aberta**: Incentivo para reportar problemas de segurança sem medo de represálias.
- **Melhoria Contínua**: Aprendizado com incidentes e adaptação de práticas.

Conclusão

A segurança e a proteção de dados são elementos fundamentais deste projeto de e-commerce. As medidas descritas neste documento foram projetadas para proteger os dados dos usuários, garantir a integridade do sistema e cumprir com os requisitos legais, especialmente a LGPD.

A implementação dessas medidas será integrada em todas as fases do desenvolvimento, desde o planejamento inicial até a manutenção contínua. A segurança não é apenas uma característica do sistema, mas um processo contínuo que requer atenção constante e adaptação às novas ameaças e requisitos.

Este documento serve como um guia abrangente para a implementação de segurança no e-commerce, mas deve ser revisado e atualizado regularmente para garantir que continue a refletir as melhores práticas e os requisitos legais mais recentes.