DevStream

POLÍTICA DE SEGURANÇA DA INFORMAÇÃO

N° Doc. 1

DevStream

1. Introdução

1.1. Objetivo

Esta política visa proteger integralmente o ecossistema de informação do GitLab, assegurando a confidencialidade, integridade e disponibilidade de todos os ativos, com ênfase em ambientes remotos e colaboração global. Os objetivos específicos incluem:

Proteção de Ativos Estratégicos:

1. Código-fonte:
    - Repositórios públicos/privados: Proteção via GitLab.com com autenticação obrigatória (SSO + MFA).
    - Assinaturas digitais: GPG para commits críticos (ex.: merge requests em projetos Core).
    - Proteção de branches: Restrição de push direto em main via Protected Branches.
2. Dados sensíveis:
    - Segredos de aplicação: Gerenciamento via GitLab CI/CD Variables com criptografia AES-256.
    - Dados de clientes: Armazenamento em Salesforce com criptografia em repouso e acesso via RBAC.
3. Infraestrutura crítica:
    - Pipelines CI/CD: Varredura automática de vulnerabilidades com GitLab Ultimate.
    - Clusters Kubernetes: Configuração segura via GitLab Agent for Kubernetes.

Conformidade e Governança

1. Certificações:
    - ISO 27001: Manutenção de controles documentados em GitLab Compliance.
    - SOC 2 Tipo II: Relatórios auditados disponíveis para clientes enterprise.
2. Regulamentações de privacidade:
    - GDPR/LGPD: Processos de Data Subject Access Requests (DSAR) via portal dedicado.
    - CCPA: Exclusão automatizada de dados de usuários californianos via API.

Gestão Proativa de Riscos

1. Shadow IT:
    - Monitoramento via Netskope para bloquear aplicações não autorizadas (ex.: Google Drive pessoal).
    - Migração de planilhas locais para GitLab Secure Files.
2. Redes domésticas:
    - Guia técnico para configuração de roteadores (WPA3, firewall habilitado) disponível no GitLab Handbook.

Cultura de Segurança

1. Programa de recompensas:
2. Bug Bounty com pagamentos de até $20,000 USD por vulnerabilidades críticas.

Transparência radical:

Divulgação pública de incidentes em Security Disclosures.

1.2. Escopo

Esta política abrange todos os elementos e atores envolvidos no ecossistema de operações do GitLab, incluindo:

Pessoas

- Colaboradores: Funcionários em tempo integral, parcial ou temporários, independentemente da localização geográfica.
- Contratados e freelancers: Desenvolvedores externos, consultores de segurança e parceiros técnicos.
- Fornecedores: Empresas terceirizadas que processam dados do GitLab (ex.: AWS, Google Cloud, Salesforce).
- Parceiros estratégicos: Integradores de sistemas e colaboradores de código aberto.

Ativos

1. Digitais:
    - Repositórios de código: GitLab.com, GitHub (para projetos mirror), e repositórios privados.
    - Dados de clientes: Armazenados em CRM (Salesforce), bancos de dados (PostgreSQL) e ferramentas de análise (Snowflake).
    - Comunicação: Slack (canais internos), Zoom (reuniões), e e-mails corporativos (Google Workspace).
2. Físicos:
    - Dispositivos remotos: Laptops, smartphones, USBs e HDs utilizados por colaboradores.
    - Infraestrutura de terceiros: Data centers de provedores de nuvem (ex.: AWS us-east-1).

Processos:

- Desenvolvimento: Pipelines CI/CD, merge requests, e revisões de código.
- Comercial: Cotação de preços, assinatura de contratos, e integração com sistemas de pagamento (ex.: Stripe).
- Suporte: Atendimento a clientes via Zendesk e interações em fóruns públicos.

Ambientes

- Nuvem pública: AWS, Google Cloud, e Azure para hospedagem de aplicações e armazenamento.
- Redes domésticas: Conexões Wi-Fi de colaboradores, roteadores pessoais e dispositivos IoT.
- Ambientes públicos: Cafés, coworkings e espaços compartilhados onde colaboradores trabalham.

Exceções

- Ferramentas pessoais não homologadas: Qualquer solução não aprovada pela equipe de segurança (ex.: Dropbox pessoal, CRMs "free") é proibida.
- Dispositivos não criptografados: Nenhum dispositivo pode acessar sistemas do GitLab sem criptografia de disco completo.

2. Princípios de Segurança

2.1. Confidencialidade

Garantir que informações sensíveis sejam acessadas, processadas ou compartilhadas apenas por indivíduos ou sistemas autorizados, conforme necessidades específicas de negócio.

Controles Implementados:

1. Autenticação Forte:

- SSO (Single Sign-On) via Google Workspace ou Okta, integrado a todas as ferramentas internas (ex.: GitLab.com, Salesforce).
- MFA (Autenticação Multifator) obrigatória para acesso a sistemas críticos (ex.: VPN, AWS Console, GitLab Ultimate).

2. Criptografia:

- Dados em trânsito: TLS 1.3 para comunicações (e-mails, APIs, repositórios Git).
- Dados em repouso: AES-256 em bancos de dados (PostgreSQL), buckets S3 e documentos confidenciais.

3. Controle de Acesso Granular:

- RBAC (Role-Based Access Control) no GitLab.com para definir permissões em repositórios (ex.: Developer, Maintainer, Reporter).
- Princípio do Menor Privilégio: Funcionários recebem acesso apenas ao necessário para suas funções.

Exemplos Práticos:

1. Código-fonte:

- Repositórios privados só são acessíveis após aprovação de merge requests por dois mantenedores.
- Protected Branches bloqueiam pushes diretos em branches críticos (ex.: main, production).

2. Dados de Clientes:

- Dados armazenados no Salesforce são segmentados por região (ex.: GDPR para UE, LGPD para Brasil) com acesso restrito a equipes específicas.

Cultura Organizacional:

- Transparência com Controle: Documentos estratégicos são compartilhados internamente, mas protegidos por níveis de acesso.

2.2. Integridade

Assegurar que informações e sistemas mantenham precisão, consistência e autenticidade ao longo de seu ciclo de vida, sem alterações não autorizadas.

Controles Implementados:

1. Versionamento e Assinaturas Digitais:

- Commits assinados com GPG em projetos críticos (ex.: GitLab FOSS).
- Histórico de alterações em contratos via plataformas como DocuSign (audit trail).

2. Verificação de Integridade:

- Checksums e hashes para validação de artefatos em pipelines CI/CD (ex.: imagens Docker, pacotes npm).
- Code Review Obrigatório para merge requests, com uso de Approval Rules.

3. Proteção contra Manipulação:

- Imutabilidade de Logs: Logs de auditoria armazenados em Amazon S3 com política Write Once, Read Many (WORM).

Exemplos Práticos:

1. Pipelines CI/CD:

- Varreduras de segurança (SAST/DAST) bloqueiam merges se vulnerabilidades críticas forem detectadas.

2. Dados Financeiros:

- Transações são validadas por APIs com assinatura HMAC para prevenir adulteração.

Cultura Organizacional:

- Colaboração Aberta com Segurança: Contribuições externas são incentivadas, mas validadas via CLA (Contributor License Agreement).

2.3. Disponibilidade

Garantir que sistemas e recursos de TI estejam operacionais e acessíveis quando necessário, mesmo em cenários de falha ou ataque.

Controles Implementados:

1. Redundância e Tolerância a Falhas:

- Multi-cloud Strategy: Hospedagem em AWS (us-east-1) e Google Cloud (europe-west3) com balanceamento de carga.
- Kubernetes Clusters: Configurados com auto-scaling e pods distribuídos em múltiplas zonas.

2. Monitoramento Proativo:

- GitLab Status Page: Monitoramento em tempo real de serviços (ex.: GitLab.com, CI/CD).
- SLA de 99.9%: Garantido por contratos com provedores de nuvem e equipe SRE dedicada.

3. Resiliência a Desastres:

- Backups Automatizados: Diários para bancos de dados e semanais para repositórios, armazenados em regiões geograficamente dispersas.
- DRP (Disaster Recovery Plan): Testado semestralmente com simulações de falhas em data centers.

Exemplos Práticos:

1. GitLab.com:

- Uso de GitLab Geo para replicação de repositórios em múltiplas regiões.

2. APIs Críticas:

- Rate limiting e circuit breakers para prevenir sobrecarga (ex.: API de autenticação).

Cultura Organizacional:

Transparência Radical em Incidentes: Interrupções são comunicadas publicamente em status.gitlab.com, com relatórios pós-mortem abertos.

2.4. Não Repúdio

3. Gerenciamento de Acesso

O Gerenciamento de Acesso é a espinha dorsal da segurança em uma organização remota como o GitLab. Esta seção detalha os controles para garantir que apenas usuários e sistemas autorizados interajam com recursos críticos, alinhando-se aos princípios de Zero Trust e menor privilégio.

3.1. Controle de Acesso e Identidades

Gerenciar identidades humanas e não humanas para garantir que o acesso seja concedido apenas com base em necessidades explícitas de negócio.

Implementação no GitLab:

1. Gestão Centralizada de Identidades:

    1.1. Single Sign-On (SSO):

    - Integração com Okta e Google Workspace para acesso unificado a ferramentas como GitLab.com, Salesforce e AWS.
    - Exemplo: Um desenvolvedor acessa o GitLab.com com credenciais do Google Workspace, sem necessidade de senha adicional.

    1.2. Provisionamento Automático:

    - Integração com sistemas de RH (ex.: BambooHR) para criar/desativar contas automaticamente.
    - Exemplo: Ao ser contratado, um funcionário recebe acesso ao GitLab.com e Slack em 24 horas; ao deixar a empresa, seu acesso é revogado imediatamente.

2. RBAC (Role-Based Access Control):

    2.1. Papéis no GitLab.com:

    - Guest (acesso somente leitura), Reporter (acessa issues), Developer (push em branches não protegidos), Maintainer (aprova merges), Owner (gerencia configurações).
    - Exemplo: Um Reporter não pode alterar código, mas pode reportar bugs via GitLab Issues.

    2.2. Grupos e Subgrupos:

    - Organização hierárquica de projetos (ex.: gitlab-com/sec/cloud para equipe de segurança em nuvem).

3. Revisão Periódica de Acessos:

- Auditorias trimestrais via GitLab Audit Events para identificar permissões obsoletas.
- Exemplo: Acesso a buckets S3 não utilizados é revogado após 60 dias de inatividade.

3.2. Autenticação

Verificar a identidade de usuários e sistemas antes de conceder acesso, mitigando riscos de credenciais comprometidas.

Implementação no GitLab:

1. Autenticação Multifator (MFA):

- Obrigatória para:
  - Acesso ao VPN (WireGuard), GitLab Ultimate, AWS Console e Google Workspace.
- Métodos Aceitos:
  - YubiKey (FIDO2): Priorizado para equipes de segurança e engenharia.
  - Google Authenticator (TOTP): Para outros colaboradores.
  - Backup Codes: Armazenados em cofre físico para casos de emergência.

2. Certificados Digitais:

- SSH/GPG Keys:
  - Exigidas para commits em repositórios críticos (ex.: GitLab Runner).
  - Exemplo: Um Maintainer assina commits com GPG para validar autenticidade.

3. Autenticação Adaptativa:

- Context-Aware Access:
  - Bloqueio de logins de regiões incomuns (ex.: login da UE para usuário baseado no Brasil).
  - Exemplo: Acesso a partir de redes públicas requer autenticação adicional via SMS.

3.3. Autorização

Garantir que os usuários autorizados tenham permissões apropriadas para acessar recursos e sistemas de informação e, assim, possam desempenhar suas funções apropriadamente.

Implementação no GitLab:

1. Princípio do Menor Privilégio:

- GitLab Protected Branches:
  - Branches como main e production exigem merge requests aprovados por dois Maintainers.
  - Exemplo: Um Developer não pode fazer push direto em main, apenas em branches de feature.
- AWS IAM Policies:
  - Permissões granulares (ex.: s3:GetObject apenas para buckets específicos).

2. Aprovações em Duas Etapas:

- Merge Requests Críticos:
  - Requerem aprovação de pelo menos dois Maintainers e passam por varredura de segurança (SAST/DAST).
- Acesso Privilegiado:
  - Solicitação via GitLab Access Requests com justificativa e prazo definido.

4. Segurança de Redes e Comunicações

A segurança de redes e comunicações é essencial para proteger a infraestrutura distribuída do GitLab, garantindo que dados e sistemas estejam protegidos contra ameaças internas e externas em um ambiente 100% remoto.

4.1. Proteção de redes

Implementar controles para mitigar riscos associados a redes não confiáveis, ataques externos e configurações inadequadas.

Controles Implementados:

1. Segmentação de Rede:

- VPCs (Virtual Private Clouds) na AWS e GCP com sub-redes isoladas para produção, desenvolvimento e testes.
- Exemplo: Dados sensíveis (ex.: tokens de API) são armazenados em sub-redes sem acesso à internet pública.

2. VPN Corporativa:

- WireGuard configurado com MFA para acesso a recursos internos (ex.: bancos de dados, ferramentas de monitoramento).
- Exemplo: Colaboradores conectam-se à VPN apenas para acessar sistemas críticos, seguindo o princípio Zero Trust.

3. Firewalls e Grupos de Segurança:

- Regras restritivas em nuvem (ex.: AWS Security Groups, GCP Firewall Rules) permitindo apenas tráfego essencial.
- Exemplo: Porta 22 (SSH) bloqueada para IPs externos, exceto para administradores autorizados via VPN.

4. Proteção contra DDoS:

- Uso de AWS Shield Advanced e Google Cloud Armor para mitigar ataques de negação de serviço.

5. Configuração de Redes Domésticas:

- Guia para colaboradores configurar roteadores com WPA3, firewall habilitado e atualizações de firmware.

4.2. Monitoramento e detecção de intrusões

Estabelecer sistemas de monitoramento e detecção de intrusões para identificar e responder a incidentes de segurança.

Controles Implementados:

1. SIEM (Security Information and Event Management):

- Splunk e Datadog para agregação e correlação de logs de sistemas, redes e aplicações.
- Exemplo: Alertas automáticos para múltiplas tentativas de login falhadas em contas privilegiadas.

2. IDS/IPS (Sistemas de Detecção/Prevenção de Intrusões):

- Suricata e AWS GuardDuty para detectar padrões de tráfego suspeitos (ex.: port scanning, exploração de vulnerabilidades).

3. Análise de Tráfego em Tempo Real:

- Uso de Zeek (Bro) para inspecionar fluxos de rede e identificar anomalias.

4. Threat Intelligence:

- Integração com feeds de ameaças (ex.: AlienVault OTX, MITRE ATT&CK) para atualizar regras de detecção.

5 Segurança em Desenvolvimento de Software (DevSecOps)

5.1. Integração de Segurança no Ciclo de Vida (SDLC)

A segurança é integrada desde o planejamento até a produção, seguindo o princípio "Shift Left".

1. Planejamento e Design:

- Modelagem de Ameaças:
  - Uso de frameworks como OWASP Threat Dragon para identificar riscos em novas funcionalidades.
  - Exemplo: Análise de ameaças para a integração de autenticação SAML no GitLab.com.
- Requisitos de Segurança:
  - Critérios como a "criptografia de dados em trânsito" são incluídos em issues e epics.

2. Desenvolvimento:

- Revisão de Código com Foco em Segurança:
  - Merge Requests exigem aprovação de pelo menos um Maintainer com expertise em segurança.
  - Exemplo: Uso de Code Owners para definir responsáveis por arquivos críticos.
- Prevenção de Vazamento de Segredos:
  - GitLab Secret Detection escaneia commits para identificar tokens, chaves API ou credenciais expostas.

3. Testes Automatizados:

- Pipelines CI/CD com Etapas de Segurança:
  - Execução de SAST, DAST, Dependency Scanning e Container Scanning em estágios dedicados.
- Testes de Container:
  - Verificação de imagens Docker com Trivy para detectar CVEs.

4. Deploy e Monitoramento:

- Implantação Gradual com Feature Flags:
  - Novas funcionalidades são liberadas gradualmente para mitigar riscos.
- Monitoramento Contínuo:
  - GitLab Threat Monitoring identifica anomalias em APIs e aplicações.

5.2. Uso de Ferramentas de Varredura Automática

Utilizar ferramentas automáticas como SAST, DAST, Dependency Scanning e Container Scanning, executadas nas pipelines CI/CD. Vulnerabilidades críticas bloqueiam o avanço dos projetos até serem resolvidas.

Ferramentas e Processos:

1. SAST (Static Application Security Testing):

- Analisa código-fonte em busca de vulnerabilidades como SQLi, XSS e buffer overflows.
- Integração no GitLab:
  - Relatórios gerados automaticamente em Merge Requests (ex.: Flawfinder para C/C++).

2. DAST (Dynamic Application Security Testing):

- Testa aplicações em execução (ex.: ambientes de staging) para identificar falhas como CSRF e SSRF.
- Exemplo: Configuração de DAST para APIs REST usando GitLab DAST.

3. Dependency Scanning:

- Detecta dependências vulneráveis em package managers (ex.: npm, pip, Maven).
- Bloqueio de Merge Requests: Se uma dependência com CVSS ≥ 9.0 for detectada, o pipeline falha e bloqueia o 'merge'.

4. Container Scanning:

- Verifica imagens Docker/OCI para identificar pacotes desatualizados ou mal configurados.
- Integração com Kubernetes: Imagens não aprovadas são bloqueadas no cluster via OPA Gatekeeper.

5.3. Gestão de Dependências e Licenças

Gerenciar dependências e licenças com varreduras automáticas e políticas de aprovação rigorosas. Dependências vulneráveis ou com licenças incompatíveis são detectadas e bloqueadas automaticamente para garantir conformidade e segurança.

Processos e Ferramentas:

1. Dependency Scanning Automatizado:

- Ferramentas: GitLab Dependency Scanning integrado a scanners como Gemnasium.
- Ações: Atualizações automáticas via Dependabot ou Renovate.

2. Gestão de Licenças:

- License Compliance:
  - Varredura de licenças em dependências (ex.: GPL, MIT) e bloqueio de licenças proibidas (ex.: AGPL).
- Políticas Customizáveis:
  - Lista de licenças permitidas configurável via .gitlab-license-management.yml.

3. Conformidade com Padrões:

- Relatórios de licenças gerados para auditorias (ex.: SOC 2, ISO 27001).
- Integração com SPDX para padronização de metadados.

6. Gestão de Incidentes de Segurança

6.1. Resposta a incidentes

Conter, erradicar e recuperar sistemas afetados, além de aprender com incidentes para prevenir recorrências.

Processo de Resposta:

1. Preparação:

- Equipe de Resposta (SIRT):
  - Composta por especialistas em segurança, engenharia, jurídico e comunicação.
  - Membros são acionados via PagerDuty em caso de incidentes críticos.

- Playbooks Documentados:
    - Procedimentos específicos para cenários como ransomware, vazamento de dados ou comprometimento de código.
    - Exemplo: Playbook para Vazamento de Tokens.

2. Identificação:

- Detecção:
    - Alertas via SIEM (Splunk), IDS/IPS (Suricata) ou relatórios de funcionários.
- Classificação:
    - Severidade baseada em critérios como impacto financeiro, exposição de dados e tempo de resolução (ex.: Crítico, Alto, Médio, Baixo).

3. Contenção:

- Ações Imediatas:
    - Isolar sistemas afetados (ex.: revogar tokens comprometidos, bloquear IPs maliciosos).
    - Exemplo: Bloqueio de uma conta AWS comprometida via IAM Policy.

4. Eradicação e Recuperação:

- Remoção de Ameaças:
    - Eliminar malware, corrigir vulnerabilidades e restaurar backups válidos.
- Validação:
    - Testes de segurança pós-recuperação para garantir que a ameaça foi eliminada.

5. Lições Aprendidas (Post-Mortem):

- Relatório Público:
    - Divulgação detalhada em Security Disclosures com causas raiz e ações corretivas.
- Atualização de Políticas:
    - Revisão de controles para prevenir recorrências (ex.: melhoria de MFA após phishing bem-sucedido).

6.2. Relatórios de incidentes

Os funcionários devem relatar todos os incidentes de segurança imediatamente à equipe de segurança da informação.

Procedimentos de Reporte:

1. Canais de Reporte:

- Formulário de Segurança: Formulário de Divulgação para colaboradores e terceiros.
- HackerOne: Para reportes externos via Programa de Bug Bounty.
- Slack: Mensagem direta nos canais internos da empresa à equipe de segurança apropriada.

2. Informações Obrigatórias:

- Descrição do incidente, sistemas afetados, horário e evidências (ex.: logs, capturas de tela).

3. Triagem Inicial:

- Classificação da criticidade pela equipe SIRT em até 1 hora após o reporte.

- Comunicação ao CEO e diretoria se classificado como Crítico (ex.: violação de dados de clientes).

Treinamento e Conscientização:

- Simulações de Incidentes: Exercícios trimestrais (ex.: ataque de ransomware simulado) para testar a prontidão.
- Orientação para Funcionários:
    - Como identificar incidentes (ex.: e-mails de phishing, comportamentos anômalos em sistemas).
    - Não apagar evidências (ex.: logs, dispositivos afetados).

7. Gestão de Terceiros e Fornecedores

7.1. Avaliação de Riscos de Terceiros

Realizar avaliações rigorosas de risco antes de contratar terceiros, analisando práticas de segurança, compliance e proteção de dados para garantir alinhamento com nossos padrões de segurança.

Processo de Avaliação:

1. Critérios de Seleção:

- Certificações de Segurança: Exigência de ISO 27001, SOC 2 Tipo II ou equivalentes.
- Práticas de Proteção de Dados: Criptografia de dados em trânsito/repouso e adesão ao GDPR/LGPD.
- Histórico de Incidentes: Análise de violações passadas e capacidade de resposta.

2. Avaliação Técnica:

- Questionários Padronizados: Uso de CAIQ (Consensus Assessments Initiative Questionnaire) para avaliar controles de segurança.
- Due Diligence: Revisão de arquitetura de sistemas, políticas de acesso e práticas de desenvolvimento seguro.

3. Classificação de Risco:

- Níveis de Criticidade:
    - Alto Risco: Fornecedores com acesso a dados sensíveis (ex.: AWS, Salesforce).
    - Médio Risco: Parceiros de suporte técnico sem acesso direto a dados.
    - Baixo Risco: Fornecedores de serviços não críticos (ex.: ferramentas de produtividade).

7.2. Contratos e SLAs de Segurança

Todos os contratos do GitLab com fornecedores incluem cláusulas específicas de segurança da informação, confidencialidade, proteção de dados e SLAs (Acordos de Nível de Serviço) para incidentes e atualizações de segurança.

Cláusulas Essenciais:

1. Proteção de Dados:

- DPA (Data Processing Agreement): Alinhado ao GDPR, com definição de papéis (controlador/processador).
- Criptografia: Exigência de AES-256 para dados em repouso e TLS 1.3 para trânsito.

2. Resposta a Incidentes:

- Notificação de Violações: Prazo máximo de 24 horas após detecção.
- Cooperação em Investigação: Suporte técnico e legal durante crises.

3. SLAs de Segurança:

- Atualizações de Vulnerabilidades: Correção de falhas críticas em até 7 dias.
- Tempo de Resposta:
    - Crítico: 1 hora (ex.: vazamento de dados).
    - Alto: 4 horas (ex.: falha de autenticação).

7.3. Monitoramento Contínuo de Fornecedores

Realizar monitoramento contínuo de seus fornecedores, incluindo revisões de conformidade, auditorias periódicas e reavaliações de risco para garantir que padrões de segurança sejam mantidos ao longo da relação contratual.

Processos de Monitoramento:

1. Avaliações Periódicas:

- Auditorias Anuais: Realizadas por equipe interna ou terceira parte (ex.: verificação de certificações).
- Relatórios de Conformidade: Exigência de relatórios trimestrais de fornecedores de alto risco (ex.: logs de acesso, testes de penetração).

2. Indicadores de Desempenho (KPIs):

- Security Score: Pontuação baseada em ferramentas como SecurityScorecard ou BitSight.
- Taxa de Incidentes: Número de violações relacionadas ao fornecedor nos últimos 12 meses.

3. Ações Corretivas:

- Planos de Melhoria: Fornecedores com não conformidades devem apresentar plano em 15 dias.
- Término de Contrato: Em caso de falhas repetidas ou violações graves (ex.: negligência com dados de clientes).

8. Conscientização e Treinamento em Segurança

A conscientização e o treinamento em segurança são pilares fundamentais para manter uma cultura de segurança proativa no GitLab. Em um ambiente 100% remoto, é essencial que todos os colaboradores compreendam seu papel na proteção dos ativos da organização e estejam atualizados sobre ameaças emergentes.

8.1. Programa de conscientização

Desenvolver e implementar um programa de conscientização sobre segurança para garantir que todos os funcionários compreendam suas responsabilidades e a importância da segurança da informação.

Controles Implementados:

1. Ciclo de Conscientização Contínua:

- Conteúdo Mensal: E-mails, posts no Slack e artigos no GitLab Handbook sobre temas como phishing, engenharia social e proteção de dispositivos.

- Simulações de Phishing: Campanhas trimestrais com cenários realistas (ex.: e-mails falsos de "atualização de senha").
    - Métricas: Taxa de cliques, reportes de incidentes e feedback dos colaboradores.

2. Eventos e Workshops:

- Security Awareness Month: Palestras com especialistas, competições internas (ex.: CTF Capture the Flag) e premiações para equipes mais engajadas.
- Coffee Chats: Sessões informais no Zoom para discutir tópicos como segurança em redes domésticas ou uso de VPN.

3. Recursos Personalizados:

- Guia de Segurança para Trabalhadores Remotos: Instruções sobre configuração de roteadores, uso de Wi-Fi público e proteção de dispositivos pessoais.
- Checklists Interativos: Ferramentas no GitLab Handbook para verificação de práticas seguras.

8.2. Treinamento em segurança

Fornecer treinamento em segurança regularmente para manter os funcionários atualizados sobre as melhores práticas e políticas de segurança.

Programas de Treinamento:

1. Treinamentos Obrigatórios:

- Onboarding: Curso introdutório de 2 horas sobre políticas de segurança, uso de MFA e proteção de dados.
- Anual: Atualização sobre novas ameaças (ex.: ransomware, deepfakes) e mudanças nas políticas.

2. Treinamentos Especializados por Função:

- Desenvolvedores:
    - Secure Coding: Práticas para evitar vulnerabilidades (ex.: SQLi, XSS) usando GitLab Secure Code Training.
    - DevSecOps: Integração de SAST/DAST em pipelines CI/CD.
- Equipe de TI:
    - Resposta a Incidentes: Uso de ferramentas como Splunk e Wazuh.
- Liderança:
    - Gestão de Riscos: Workshops sobre decisões estratégicas em segurança.

3. Formatos Diversificados:

- Cursos Online: Parceria com plataformas como Pluralsight e Coursera.
- Hands-On Labs: Ambientes simulados para praticar resposta a incidentes (ex.: ataque a um bucket S3 exposto).

9. Continuidade de Negócios e Recuperação de Desastres (BC/DR)

9.1. Planos de Continuidade de Negócios

Mantém planos formais de continuidade que garantem a operação dos serviços críticos mesmo durante falhas, desastres naturais ou ataques cibernéticos.

Controles Implementados:

1. Identificação de Serviços Críticos:

- Classificação por Prioridade:
  - Nível 1: GitLab.com, CI/CD pipelines, autenticação SSO.
  - Nível 2: Ferramentas internas (Salesforce, Zendesk).
- RTO (Recovery Time Objective):
  - Serviços críticos: ≤ 4 horas.
  - Serviços não críticos: ≤ 24 horas.
- RPO (Recovery Point Objective):
  - Dados críticos: ≤ 15 minutos de perda.

2. Estratégias de Redundância:

- Multi-cloud: Hospedagem em AWS (us-east-1) e Google Cloud (europe-west3) com balanceamento de carga.
- GitLab Geo: Replicação de repositórios em regiões geograficamente dispersas.

3. Plano de Continuidade Documentado:

- Procedimentos detalhados no GitLab Runbooks, incluindo:
  - Ativação de ambientes alternativos.
  - Comunicação com clientes via Status Page.

9.2. Backup e Restauração de Dados

Realizar backups automáticos e regulares de dados, com armazenamento seguro em múltiplas regiões, garantindo rápida recuperação em caso de perda.

Controles Implementados:

1. Política de Backup:

- Frequência:
  - Bancos de dados: Backups incrementais a cada 15 minutos e completos diários.
  - Repositórios Git: Backup contínuo via GitLab Geo.
- Armazenamento:
  - Backups criptografados (AES-256) em AWS S3 Glacier e Google Cloud Storage, em múltiplas regiões.
- Retenção:
  - 30 dias para backups diários.
  - 1 ano para backups mensais.

2. Restauração Automatizada:

- Scripts Validados: Recuperação de bancos de dados PostgreSQL em ≤ 30 minutos.
- Testes de Integridade: Verificação semanal de backups via checksums.

9.3. Testes de Recuperação de Desastres

Realizar testes periódicos de recuperação de desastres para validar a eficácia dos planos, ajustar processos e assegurar o tempo mínimo de indisponibilidade em eventos críticos.

Processos de Teste:

1. Simulações Programadas:

- Testes Anuais: Simulação de desastre total (ex.: perda simultânea de duas regiões de nuvem).
- Testes Trimestrais: Failover parcial (ex.: migração de tráfego entre AWS e GCP).

2. Cenários de Teste:

- Ataque Ransomware: Recuperação de sistemas a partir de backups imutáveis.
- Falha de Data Center: Ativação de ambientes secundários via Terraform.

3. Pós-Teste:

- Relatório de Lições Aprendidas.
- Atualização de Playbooks: Ajustes em procedimentos com base em gaps identificados.

10. Gestão de Mudanças

10.1. Processo Formal de Aprovação de Mudanças

Adoção de processo formal de gestão de mudanças, exigindo revisões, aprovações e validações antes da implementação de qualquer alteração em ambientes de produção.

Etapas do Processo:

1. Solicitação de Mudança:

- Registrada via GitLab Issues ou ServiceNow, com detalhes como justificativa, impacto e plano de reversão.

2. Análise de Risco:

- Avaliação técnica por engenheiros e equipe de segurança (ex.: impacto em SLA, exposição de dados).

3. Aprovação:

- Change Advisory Board (CAB): Comitê com representantes de segurança, engenharia e operações.
- Aprovação em Duas Etapas: Merge Requests exigem aprovação de um Maintainer e um revisor de segurança.

4. Implementação:

- Janelas de mudança pré-definidas (ex.: fora do horário comercial para serviços críticos).
- Uso de Feature Flags para liberação gradual.

10.2. Impacto em Segurança de Atualizações

Todas as mudanças são avaliadas quanto ao impacto na segurança, com análises específicas para vulnerabilidades, compatibilidade e proteção de dados.

Avaliações de Segurança:

1. Análise de Vulnerabilidades:

- SAST/DAST: Varreduras automáticas em pipelines CI/CD para mudanças de código.
- Dependency Scanning: Verificação de atualizações de bibliotecas (ex.: CVE em log4j).

2. Modelagem de Ameaças:

- Sessões colaborativas usando OWASP Threat Dragon para mudanças críticas (ex.: nova integração de API).

3. Revisão de Configurações:

- Verificação de permissões (ex.: IAM roles na AWS) e exposição de endpoints.

10.3. Registro e Auditoria de Mudanças

Manter registros detalhados de todas as mudanças realizadas, com trilhas de auditoria para assegurar rastreabilidade, conformidade e investigação rápida em caso de incidentes.

Controles Implementados:

1. Registros Detalhados:

- GitLab Audit Events: Logs de alterações em permissões, 'merge requests' e configurações.
- Infraestrutura como Código (IaC): Histórico de mudanças em arquivos Terraform/Ansible via Git.

2. Imutabilidade de Logs:

- Armazenamento em Amazon S3 com política WORM (Write Once, Read Many) para prevenir adulteração.

3. Auditorias Periódicas:

- Revisões mensais por equipe de compliance para verificar conformidade com políticas internas e regulatórias.

11. Uso Aceitável de Recursos de TI

11.1. Política de Uso de Dispositivos Pessoais (BYOD)

Permissão de uso de alguns dispositivos pessoais desde que eles tenham requisitos de segurança, como senha e 2FA, mas proíbe o acesso a dados sensíveis. Já para a comunicação dentro da empresa é obrigatório o uso do slack e Google Drive.

1. Requisitos Mínimos de Segurança:

- Senhas Fortes: Exigência de senhas com 12+ caracteres (incluindo números, símbolos e letras maiúsculas/minúsculas).
- Autenticação Multifator (MFA): Obrigatória para acesso a sistemas corporativos (ex.: VPN, GitLab.com).
- Criptografia de Disco: BitLocker (Windows) ou FileVault (macOS) ativados.

2. Restrições de Acesso:

- Dados Sensíveis Proibidos: Dispositivos pessoais não podem armazenar dados classificados como Confidenciais ou Restritos (ex.: tokens de API, chaves SSH).
- Acesso via Navegador Seguro: Dados sensíveis só podem ser acessados via navegadores atualizados (ex.: Chrome, Firefox) com extensões bloqueadas.

3. Ferramentas Obrigatórias para Comunicação:

- Slack: Único canal autorizado para discussões internas (com criptografia de ponta a ponta em canais privados).
- Google Drive Corporativo: Armazenamento exclusivo para documentos da empresa (com políticas de acesso baseadas em roles).

11.2. Restrições a Aplicações Não Homologadas

Software de Uso Individual (freeware, complementos e plugins) é permitido, com exceção de integrações não autorizadas do Google Workspace e extensões do Chrome. O Software de Uso Individual está sujeito à remoção retroativa a qualquer momento pelo Departamento de TI, Departamento Jurídico e Segurança quando for considerado inseguro ou inseguro.

Políticas Específicas:

1. Permitido com Restrições:

- Freeware/Plugins: Uso permitido após revisão rápida pelo time de segurança (ex.: ferramentas de produtividade como Notion).
- Extensões de Navegador: Apenas as aprovadas pela equipe de TI (ex.: Grammarly, LastPass).

2. Proibições Explícitas:

- Integrações Não Autorizadas no Google Workspace: Ex.: Apps que solicitam acesso a e-mails ou Drive sem aprovação.
- Extensões do Chrome Não Validadas: Bloqueadas via política corporativa.

3. Remoção Retroativa:

- Critérios para Remoção:
  - Vulnerabilidades críticas (CVSS ≥ 7.0).
  - Violação de licenças ou termos de uso.
- Processo: Notificação prévia ao usuário e desinstalação remota via MDM.

11.3. Monitoramento de Atividades Suspeitas

Monitorar constantemente atividades em nossa infraestrutura para identificar comportamentos suspeitos e agir rapidamente em caso de incidentes.

Controles Implementados:

1. Monitoramento Contínuo:

- SIEM (Splunk): Agregação de logs de sistemas, redes e aplicações para correlação de eventos.
- IDS/IPS (Suricata): Detecção de padrões de ataque (ex.: port scanning, SQL injection).

2. Alertas Automatizados:

- Comportamentos de Risco:
  - Múltiplas tentativas de login falhadas.
  - Acesso a dados sensíveis fora do horário habitual.
- Integração com PagerDuty: Notificação imediata da equipe de segurança.

3. Resposta Rápida:

- Playbooks de Resposta:
    - Bloqueio automático de IPs maliciosos via AWS WAF.
    - Revogação de tokens comprometidos via HashiCorp Vault.

12. Gestão de Identidades e Acesso Privilegiado (PAM)

12.1 Controle de Contas Privilegiadas

Seguir o princípio do menor privilégio de modo que o acesso privilegiado só é concedido com aprovações formais e revisões periódicas

Controles Implementados:

1. Processo de Aprovação Formal:

- Solicitações de acesso privilegiado são submetidas via GitLab Access Requests com justificativa técnica e prazo definido.
- Aprovação requerida por pelo menos dois líderes: um técnico (ex.: Engenheiro de Sênior) e um de segurança.

2. Revisões Periódicas:

- Auditorias mensais de contas com acesso a sistemas críticos (ex.: bancos de dados, servidores de produção).
- Exemplo: Acesso root a servidores AWS é revogado após 30 dias se não houver necessidade comprovada.

3. Segregação de Funções (SoD):

- Separação entre desenvolvedores e administradores de infraestrutura para evitar conflitos de interesse.

12.2. Rotação de Credenciais

As credenciais de contas privilegiadas são rotacionadas regularmente ou, em casos de incidentes, imediatamente.Além disso o GitLab também permite que os usuários realizem a detecção de segredos para verificar segredos e credenciais comprometidos involuntariamente. Os usuários do GitLab Ultimate podem aplicar respostas automáticas a segredos vazados , como revogar o segredo, para mitigar o impacto do vazamento de credenciais.

Controles Implementados:

1. Rotação Automatizada:

- Credenciais de Infraestrutura: Chaves AWS, tokens de API e senhas de banco de dados são rotacionadas a cada 90 dias via HashiCorp Vault.
- Integração com GitLab CI/CD: Pipelines atualizam segredos em ambientes de produção sem intervenção manual.

2. Detecção de Segredos Expostos:

- GitLab Secret Detection: Varre automaticamente repositórios em busca de tokens, chaves ou credenciais expostas.
- Exemplo: Se uma chave AWS é encontrada em um commit, o pipeline bloqueia o merge request e notifica a equipe de segurança.

3. Resposta Automática a Vazamentos:

- GitLab Ultimate:

- Revogação automática de segredos vazados via integração com ferramentas como AWS IAM ou GitHub.
- Exemplo: Um token OAuth exposto no GitHub é revogado em 5 minutos após detecção.

12.3. Monitoramento de Sessões Privilegiadas

As Sessões Privilegiadas são constantemente monitoradas e auditadas de modo a garantir rastreabilidade e proteção contra o uso indevido.

Controles Implementados:

1. Gravação de Sessões:

- SSH e RDP: Todas as sessões remotas são gravadas e armazenadas em Amazon S3 com criptografia AES-256.
- Ferramentas: Teleport para acesso seguro a servidores e auditoria de sessões.

2. Alertas em Tempo Real:

- Comportamentos Anômalos: Ações como execução de comandos privilegiados (ex.: sudo rm -rf /) disparam alertas no PagerDuty.
- Integração com SIEM: Logs de sessões são correlacionados com outros eventos no Splunk para detectar padrões suspeitos.

3. Revisão Pós-Ação:

- Auditorias Semanais: Equipe de segurança revisa 10% das sessões privilegiadas aleatoriamente.
- Exemplo: Uma sessão SSH às 3 AM em um servidor de produção é investigada para verificar legitimidade.

13. Avaliação e Melhoria Contínua

13.1. Auditorias de segurança

Realizar auditorias de segurança periódicas para avaliar a eficácia das políticas e práticas de segurança e identificar áreas de melhoria.

Controles Implementados:

1. Auditorias Internas:

- Trimestrais: Realizadas pela equipe de segurança interna, focando em áreas críticas (ex.: acesso privilegiado, proteção de dados).
- Ferramentas: Uso de GitLab Compliance Dashboard para automatizar verificações de conformidade em pipelines CI/CD.

2. Auditorias Externas:

- Anuais: Realizadas por empresas terceirizadas (ex.: Ernst & Young) para certificações como ISO 27001, SOC 2 Tipo II e GDPR.
- Pentests Contratados: Testes de invasão semestrais por empresas como Cure53 ou HackerOne.

3. Automatização de Auditorias:

- Infraestrutura como Código (IaC): Verificação contínua de configurações seguras via Terraform e Ansible.

- Exemplo: Pipelines CI/CD executam verificações de compliance usando InSpec.

13.2. Revisão de políticas e procedimentos

Revisar e atualizar as políticas e procedimentos de segurança regularmente, considerando as mudanças tecnológicas, as ameaças emergentes e as lições aprendidas com incidentes de segurança anteriores.

Processo de Revisão:

1. Ciclo de Revisão:

- Semestral: Revisão formal de todas as políticas de segurança.
- Atualizações Ad Hoc: Em resposta a incidentes críticos ou novas regulamentações (ex.: LGPD, CCPA).

2. Colaboração Aberta:

- Contribuições da Comunidade: Funcionários e colaboradores externos podem sugerir mudanças via Merge Requests no Handbook do GitLab.

3. Documentação Transparente:

- Histórico de alterações disponível publicamente no GitLab Changelog.

13.3. Análise de riscos

Realizar análises de risco para identificar e avaliar os riscos de segurança associados às informações e ativos da organização, e implementar medidas para mitigar esses riscos.

Metodologia:

1. Framework Utilizado:

- NIST SP 800-30: Para avaliação qualitativa de riscos.
- ISO 27005: Integrado ao Sistema de Gestão de Segurança da Informação (SGSI).

2. Processo de Avaliação:

- Identificação de Ativos: Mapeamento de sistemas críticos (ex.: GitLab.com, bancos de dados).
- Matriz de Riscos: Classificação por probabilidade e impacto (ex.: risco de vazamento de tokens: probabilidade média, impacto alto).

3. Mitigação:

- Plano de Ação: Atribuição de responsáveis e prazos para tratamento de riscos.
- Exemplo: Migração para autenticação sem senha (FIDO2) após análise de riscos de phishing.

13.4. Medição de desempenho

Estabelecer métricas e indicadores-chave de desempenho (KPIs) para medir a eficácia dos programas de segurança e garantir que os objetivos de segurança sejam alcançados.

Indicador	**Meta**	**Ferramenta de Medição**
Tempo Médio de Resposta (MTTR)	≤ 2 horas para incidentes críticos	PagerDuty / GitLab Incident

		Management
Adoção de MFA	100% dos colaboradores	Okta/Google Workspace
Conformidade em Patches	≥ 95% dos sistemas atualizados	Qualys/GitLab Vulnerability Report
Taxa de Detecção de Ameaças	≥ 90% de cobertura em SAST/DAST	GitLab Security Dashboard

14. Conformidade Legal e Regulatória

14.1. Conformidade com leis e regulamentações

Garantir que as políticas e práticas de segurança estejam em conformidade com as leis e regulamentações aplicáveis, incluindo leis de privacidade e proteção de dados.

Controles Implementados:

1. Mapeamento de Requisitos Legais:

- Adequação às Regulamentações-Chave:
  - Marco Civil da Internet (Lei n° 12.965/2014)
  - LGPD (Lei Geral de Proteção de Dados - Lei n° 13.709/2018)
  - Lei de Crimes Cibernéticos (Lei n° 12.737/2012)
  - Lei do Software (Lei n° 9.609/1998)
  - Regulamentação do Teletrabalho (Lei n° 14.442/2022)
- Certificações: Manutenção de ISO 27001, SOC 2 Tipo II e PCI DSS (para pagamentos).

2. Ferramentas e Documentação:

- GitLab Compliance Dashboard: Monitoramento centralizado de controles de conformidade em tempo real.
- Relatórios Públicos: Disponíveis em Trust Center para clientes e auditores.

14.2. Gerenciamento de vulnerabilidades e patches

Implementação de processo para identificar, avaliar e corrigir vulnerabilidades de segurança nos sistemas e aplicativos, incluindo a aplicação regular de patches de segurança.

Processos e Ferramentas:

1. Identificação de Vulnerabilidades:

- Varreduras Automatizadas:
  - SAST/DAST: Integrados a pipelines CI/CD para análise de código e aplicações em tempo real.
  - Dependency Scanning: Detecção de CVEs em dependências via GitLab Ultimate.
- Bug Bounty Program: Parceria com HackerOne para reportes externos.

2. Priorização e Correção:

- Classificação por Severidade: Baseada em CVSS (ex.: Crítico ≥ 9.0, Alto ≥ 7.0).
- SLAs de Correção:
  - Críticas: 7 dias.
  - Altas: 14 dias.

- Automação de Patches:
    - GitLab Auto DevOps: Aplicação automática de patches em ambientes de staging.
    - Renovate Bot: Atualização automatizada de dependências em repositórios.

3. Resposta a Vazamentos:

- Revogação Automática: Segredos expostos (ex.: tokens API) são revogados via integração com ferramentas como AWS IAM ou HashiCorp Vault.
- GitLab Secret Detection: Bloqueia merge requests com credenciais expostas.

15. Responsabilidades

15.1. Direção

A direção da organização é responsável por estabelecer e apoiar as políticas de segurança da informação e garantir que os recursos adequados sejam alocados para a gestão da segurança.

Responsabilidades:

1. Estabelecimento de Políticas:

- Definir e aprovar políticas de segurança, garantindo alinhamento com objetivos estratégicos e regulamentações (ex.: GDPR, ISO 27001).
- Revisar e atualizar formalmente as políticas anualmente, ou conforme mudanças tecnológicas/regulatórias.

2. Alocação de Recursos:

- Garantir orçamento adequado para ferramentas de segurança, treinamentos e resposta a incidentes (ex.: aquisição de soluções como GitLab Ultimate, contratação de pentesters).

3. Promoção de Cultura de Segurança:

- Liderar iniciativas como o Security Awareness Month e incentivar a adoção de práticas Security First em todas as equipes.

4. Prestação de Contas:

- Reportar publicamente o status de segurança aos stakeholders via Relatório Anual de Transparência.

15.2. Equipe de segurança da informação

A equipe de segurança da informação é responsável pela implementação, monitoramento e manutenção das políticas e práticas de segurança e pela resposta a incidentes de segurança.

Responsabilidades:

1. Implementação de Controles:

- Configurar e manter ferramentas como SIEM (Splunk), IDS/IPS (Suricata) e plataformas de autenticação (Okta).
- Gerenciar o GitLab Security Dashboard para monitoramento de vulnerabilidades.

2. Resposta a Incidentes:

- Atuar como primeira resposta a violações, seguindo o Playbook de Resposta a Incidentes.
- Coordenar comunicações transparentes via Status Page em caso de interrupções críticas.

3. Conformidade e Auditoria:

- Realizar auditorias internas trimestrais e preparar documentação para auditorias externas (ex.: SOC 2, ISO 27001).

4. Inovação em Segurança:

- Integrar práticas de DevSecOps (ex.: SAST/DAST automatizados) em pipelines CI/CD.

15.3. Funcionários

Todos os funcionários são responsáveis por seguir as políticas e práticas de segurança estabelecidas e por relatar qualquer incidente de segurança ou preocupações relacionadas à segurança da informação.

Responsabilidades:

1. Adesão às Políticas:

- Seguir políticas de uso aceitável (ex.: proibição de Shadow IT), proteção de dispositivos e gestão de senhas.
- Participar obrigatoriamente de treinamentos anuais de segurança e simulações de phishing.

2. Reporte Proativo:

- Notificar imediatamente incidentes via Formulário de Segurança ou canal #security no Slack.
- Reportar vulnerabilidades em código ou sistemas via HackerOne.

3. Proteção de Dados:

- Garantir que dados sensíveis (ex.: tokens, credenciais) sejam armazenados apenas em plataformas aprovadas (ex.: HashiCorp Vault, Google Drive corporativo).

16. Glossário

Revisão	Referências	Responsável	Data de Execução	Resumo de Alteração
	Política de Segurança da Informação	Nome do Diretor de Segurança	21/05/2025 Pauta da n° Reunião do Conselho Superior	Primeira Versão
1°				
2°				
3°				
4°				
5°				
6°				
7°				
8°				
9°				
10°				
11°				
12°				
13°				
14°				
15°				
16°				
17°				
18°