PILLAR

APRENDA · THE STACK

Arquitetura Zero-Secret: Como Minimizar Sua Superfície de Ataque

Cada chave de API, token ou senha que sua equipe mantém é simultaneamente uma vulnerabilidade de segurança e um custo operacional contínuo. A Pillar foi projetada desde o início para operar com o mínimo possível de segredos: aproximadamente seis no stack inteiro de produção. Este guia mostra a filosofia, a arquitetura e os passos táticos para você replicar essa abordagem.

---

9 min de leitura

Última atualização: 10 de junho de 2026

> *Quanto menos segredos você manuseia, mais seguro é o seu negócio e mais simples são suas operações. A meta não é gerenciar segredos melhor — é eliminar a necessidade deles sempre que possível.*

A Tese

- Por que cada segredo é tanto uma superfície de ataque quanto um passivo de manutenção contínua
- O framework The Zero-Secret Stack: cinco pilares para reduzir radicalmente o inventário de segredos
- Como a Pillar opera todo o seu stack de produção com aproximadamente 6 segredos contra os 20-50 de uma startup típica
- Substituições concretas: Stripe Payment Links no lugar de API keys, Formspree no lugar de SendGrid, edge functions no lugar de bancos de dados
- Como auditar seu próprio inventário de segredos hoje e definir um plano de redução trimestral

01 — O Framework: The Zero-Secret Stack

The Zero-Secret Stack

The Zero-Secret Stack é um modelo mental de cinco pilares que orienta cada decisão de arquitetura na Pillar. Em vez de perguntar "como rotacionamos esta chave?", o framework pergunta "por que ainda temos esta chave?". Cada pilar deve ser avaliado de cima para baixo — você só introduz um segredo quando todos os pilares acima foram esgotados.

1

1. Delegação de Confiança (Trust Delegation)

Mova fluxos sensíveis para serviços que assumem a responsabilidade pela segurança. Pagamentos via Stripe Payment Links significam que a Stripe lida com cartões, PCI-DSS e autenticação — você não precisa de uma chave de API porque não processa nada do seu lado. O serviço externo passa a ser o custodiante do segredo, não você.

2

2. Edge-First Computing

Use Cloudflare Workers, Pages Functions e similares para mover lógica para a borda. A borda elimina servidores SSH, credenciais de banco de dados gerenciadas por você e infraestrutura clássica que historicamente exige dezenas de segredos. O que sobra são pequenas variáveis de ambiente isoladas por Worker.

3

3. Conteúdo Estático Como Padrão (Static-by-Default)

Sites estáticos não têm banco de dados, não têm sessões, não têm runtime exposto. JSON em arquivo, build em tempo de deploy, distribuição via CDN. Você nunca precisa rotacionar uma senha de banco se não existe banco para acessar.

4

4. Formários Sem Backend (Backendless Intake)

Coleta de leads, formulários de contato e inscrições passam por serviços como Formspree, que cuidam do recebimento e entrega de e-mail. Não há SMTP credentials, não há chave de Mailgun ou SendGrid, não há servidor de e-mail. O endpoint público é gerenciado, anti-spam é gerenciado, entrega é gerenciada.

5

5. Isolamento de Segredos Inevitáveis

Quando um segredo é estritamente necessário — por exemplo, verificar assinaturas de webhooks — ele vive em um único Cloudflare Worker, em uma variável de ambiente isolada, sem acesso a outros sistemas. Um vazamento desse segredo não comprometeria seu domínio, seu provedor de pagamentos ou seu DNS.

02 — Os dados.

~6

Segredos totais no stack de produção da Pillar

AUDITORIA INTERNA PILLAR 2025

20–50

Segredos típicos em uma startup SaaS equivalente

OWASP / PESQUISAS DE MERCADO DE DEVSECOPS 2024

0

Chaves de API Stripe gerenciadas pela Pillar (Payment Links cuidam de tudo)

DOCUMENTAÇÃO STRIPE PAYMENT LINKS 2024

0

Bancos de dados em produção — sem credenciais para gerenciar

ARQUITETURA PÚBLICA PILLAR

1	0
Worker isolado para verificação de webhook da Stripe	Chaves SSH para produção — deploy via Wrangler CLI local
CLOUDFLARE WORKERS DOCUMENTAÇÃO 2024	CLOUDFLARE PAGES / WRANGLER 2024

Por que cada segredo é um passivo dobrado

A maioria das equipes pensa em segredos apenas como um problema de segurança: se vazar, alguém mal-intencionado entra. Esse é metade do problema. A outra metade é operacional: cada segredo precisa ser armazenado, rotacionado, distribuído entre membros da equipe, revogado quando alguém sai, sincronizado entre ambientes (dev, staging, prod) e auditado. Multiplique isso por 30 ou 40 segredos e você tem um trabalho de meio período apenas de gerenciamento de credenciais.

Quando um negócio cresce, esse fardo cresce de forma não-linear. Adicionar um novo desenvolvedor agora envolve dar acesso a um cofre de senhas, configurar dezenas de variáveis em ambientes locais, e treinar a pessoa em quais segredos pertencem a quais sistemas. Um único ex-funcionário com acesso a um cofre desatualizado pode arruinar meses de higiene operacional.

A Pillar opera com aproximadamente seis segredos no stack inteiro de produção. Isso não é magia — é uma série de decisões deliberadas de arquitetura, cada uma das quais examinaremos em detalhe.

Anatomia do stack: o que sobrou depois da redução

O que constitui esses ~6 segredos? Em ordem aproximada: credenciais de login da conta Cloudflare (gerenciadas pelo navegador, não pelo servidor); credenciais de login do registrador de domínio Dynadot; credenciais de login do GitHub para o repositório; um Stripe webhook secret isolado em um Cloudflare Worker; credenciais de login da Stripe Dashboard; e credenciais de e-mail administrativo. Nenhum desses é uma chave de API ativa rodando em servidores de produção — todos são credenciais humanas em painel administrativo.

O ponto crítico: nenhum servidor de produção da Pillar precisa de uma chave de API da Stripe, porque a Pillar não processa pagamentos. Os usuários clicam em um Payment Link, são redirecionados para checkout hospedado pela Stripe, completam o pagamento, e a Stripe envia um webhook assinado para um endpoint público em um Cloudflare Worker. O Worker verifica a assinatura usando o webhook secret — o único segredo de máquina em todo o stack — e dispara qualquer lógica de pós-pagamento.

Implantações seguem o mesmo padrão. Em vez de tokens de deploy de longa duração armazenados em uma plataforma de CI/CD, a Pillar usa o `wrangler` CLI rodando localmente. O Wrangler autentica via OAuth no navegador do desenvolvedor, mantendo a credencial na máquina local e não em qualquer sistema multi-tenant. Comandos como `wrangler pages deploy` ou `wrangler deploy` publicam diretamente para a Cloudflare. Quando o desenvolvedor encerra a sessão, o token expira.

Substituições concretas que você pode aplicar hoje

A pergunta não é "como copio a Pillar?" — é "quais segredos meu stack tem que eu posso eliminar?". Comece com pagamentos. Se você usa `STRIPE_SECRET_KEY` diretamente em seu backend para criar Customers, Charges ou Subscriptions programaticamente, considere se Payment Links resolvem 90% dos seus casos de uso. Para a maioria das startups vendendo planos simples ou itens únicos, eles resolvem.

E-mail transacional é outro alvo óbvio. Em vez de manter chaves de Mailgun, SendGrid ou Postmark para envio de formulários de contato, mude para Formspree ou similar. O formulário HTML envia diretamente para um endpoint hospedado, o serviço cuida da entrega de e-mail, anti-spam e rate limiting. Você remove uma chave de API do seu inventário sem perder funcionalidade.

Bancos de dados são a maior fonte única de credenciais em stacks tradicionais: connection strings, senhas de admin, tokens de replicação. Pergunte: meu site realmente precisa de um banco em runtime, ou estou usando-o para conteúdo que muda raramente? Para a maioria dos sites institucionais, ofertas SaaS de estágio inicial e landing pages, conteúdo em arquivos JSON commitado no Git, builds estáticos e CDN são suficientes. Você não precisa rotacionar uma senha de banco que não existe.

Quando segredos são inevitáveis: padrões de isolamento

Nenhum stack não-trivial chega a zero segredos absolutos. Webhooks precisam de assinaturas verificadas. APIs de terceiros às vezes exigem autenticação. Quando você precisa de um segredo, isole-o ao máximo. Na Pillar, o Stripe webhook secret vive em um único Cloudflare Worker, configurado como uma variável de ambiente criptografada (variável secreta) que não é legível via dashboard após ser definida. O comando é algo como `wrangler secret put STRIPE_WEBHOOK_SECRET`, executado uma única vez.

Esse Worker não tem acesso a DNS, a outros Workers, ao repositório, ou ao painel administrativo. Se a chave vazasse hipoteticamente, um atacante poderia, no pior caso, falsificar webhooks da Stripe para esse endpoint — e mesmo assim, a lógica downstream não concede acesso a sistemas críticos. O raio de explosão é minúsculo por design.

Regras práticas para segredos inevitáveis: nunca commite em repositório, mesmo que seja privado; armazene apenas no serviço que o consome (Cloudflare Worker environment, não em um cofre central); rotacione trimestralmente como rotina, não como reação; e mantenha um registro escrito de quais segredos existem, onde vivem, e quem tem acesso. Mesmo com 6 segredos, a documentação é importante — com 50, ela é impossível.

03 — Assista: um percurso real

04 — Checklist Tático: Auditando e Reduzindo Seu Inventário de Segredos

Você não precisa reescrever seu stack para se beneficiar deste framework. Comece com uma auditoria honesta e elimine os mais fáceis primeiro. A maioria das equipes consegue eliminar de 30% a 50% do seu inventário de segredos em um trimestre.

1. Liste todos os segredos: API keys, senhas, tokens, certificados, connection strings. Inclua os que estão em arquivos `.env` locais, em cofres como 1Password, em GitHub Secrets e em painéis de provedores.
2. Para cada segredo, anote: quem precisa dele, onde vive, quando foi rotacionado pela última vez, e qual seria o impacto se vazasse. Esse exercício sozinho geralmente revela 3-5 segredos órfãos.
3. Categorize em "substituível" vs "essencial". Pagamentos via Stripe API podem migrar para Payment Links? E-mail via SendGrid pode migrar para Formspree? Banco de dados pode virar JSON estático?
4. Para cada segredo substituível, planeje a migração em sprints curtos. Não tente refatorar tudo de uma vez — uma migração por sprint é sustentável.
5. Para segredos essenciais que sobrarem, isole-os em Cloudflare Workers (ou equivalente edge) usando `wrangler secret put <NOME>`. Um Worker, um segredo, um propósito.
6. Estabeleça um cron de rotação trimestral. Coloque no calendário. Documente o processo de rotação em um runbook simples — uma página por segredo.
7. Nunca commite segredos no repositório, nem mesmo em branches privadas. Use `git secrets` ou ferramentas equivalentes como pre-commit hook em todas as máquinas de desenvolvedores.

Perguntas frequentes.

E se eu precisar de funcionalidades que Payment Links não oferecem, como assinaturas customizadas ou planos por uso?

Payment Links suportam assinaturas recorrentes, códigos promocionais e vários fluxos comuns. Se você precisa de planos por uso medido, billing portáteis ou lógica de preços complexa, a Stripe Checkout Sessions API requer uma chave secreta. Nesse caso, isole essa chave em um único Cloudflare Worker (ou equivalente) que faz apenas a chamada para criar a sessão — não reuse essa chave em outros lugares do seu stack. Veja Zero-Secret Architecture (/learn/en/the-stack/zero-secret-architecture/) para o padrão de isolamento completo.

Sem banco de dados, como armazeno dados de usuários?

A resposta honesta: para muitos negócios de estágio inicial, você não precisa. Pagamentos são armazenados na Stripe. Inscrições são armazenadas no Formspree ou no seu provedor de e-mail. Conteúdo é commitado no Git. Quando você eventualmente precisar de um banco, use uma opção serverless edge como Cloudflare D1 ou Turso — elas se integram a Workers sem credenciais de connection string tradicionais.

Formspree não é uma dependência externa que pode falhar?

Sim — toda escolha é uma troca. A pergunta é entre "dependência em Formspree" e "dependência em Mailgun + minha própria infra de captura de formulários + monitoramento da minha infra". O segundo modelo tem mais peças que podem falhar e exige mais segredos para serem gerenciados. Formspree consolida vários riscos em um único provedor com SLA explícito.

Como faço deploys se não tenho tokens de CI/CD armazenados?

Use o CLI do seu provedor diretamente da máquina do desenvolvedor. `wrangler pages deploy` para Cloudflare Pages, `vercel` para Vercel, e assim por diante. O CLI autentica via OAuth no navegador local; o token resultante vive na máquina do desenvolvedor, não em um sistema multi-tenant. Para deploys que precisam ser automatizados (raros para startups de estágio inicial), considere OIDC do GitHub Actions, que troca confiança de máquina por confiança de identidade — eliminando o token armazenado.

Com que frequência devo rotacionar os segredos que sobraram?

Trimestralmente como rotina, imediatamente em caso de suspeita de compromisso, e sempre que alguém com acesso sair da equipe. Para segredos isolados em um único Worker, a rotação é tipicamente um único comando `wrangler secret put <NOME>` seguido de uma atualização no painel do provedor (por exemplo, gerar novo webhook secret na Stripe Dashboard). Documente o processo em um runbook de uma página por segredo.
