

IA **sem censura**, do jeito certo

Seu próprio modelo para testar a segurança do **seu** app: três rotas no DeepSeek Harness (API hospedada, GPU na RunPod, LM Studio local), ferramentas, laboratório local e custo sob controle.

INEMA.CLUB **SEGURANÇA**

IA sem censura, do jeito certo



Uso defensivo. Só sistemas seus ou com permissão por escrito do dono, dentro do escopo combinado. Credenciais fora do chat. Confira o que a IA encontrar antes de agir.

O que tem aqui

1. Comece aqui
2. Uso defensivo
3. Escolha a rota
4. Rota 1: API hospedada
5. Rota 2: GPU na RunPod
6. Rota 3: LM Studio no seu computador
7. Ferramentas
8. Segurança do modelo baixado
9. Custo da RunPod sob controle
10. Problemas comuns
11. Configurar com o seu assistente
12. O que foi testado
13. Checklist da sua instalação
14. Os 9 prompts

Os arquivos editáveis (Markdown, prompts, scripts e testes) estão no repositório. Este PDF é a versão de leitura do mesmo conteúdo.

Comece aqui

Uso defensivo. Só sistemas seus ou com permissão por escrito do dono. Credenciais fora do chat. Confira o que a IA encontrar antes de agir.

Este kit liga um modelo de IA "sem censura" (abliterated) ao **DeepSeek Harness** por uma de três rotas — API hospedada, GPU alugada na RunPod ou LM Studio no seu computador — e dá ferramentas para provar que cada parte funciona, praticar num laboratório falso e não deixar a GPU cobrando à toa.

Caminho mais curto

1. Leia **USO-DEFENSIVO.md** (5 minutos). É a parte mais importante.
2. Escolha a rota: **guias/1-escolha-da-rota.md**. Hospedada tem menos passos; local não tem conta de GPU, se a sua máquina aguentar.
3. Abra **CONFIGURA-COM-ASSISTENTE.md**, copie o pedido e entregue ao seu assistente junto com esta pasta.
4. Siga os **prompts** da sua rota, na ordem. Chave você digita direto no app ou num arquivo privado, nunca no chat.
5. Rode os testes de modelo e o laboratório antes de ligar ferramentas a qualquer coisa que importe.
6. Na RunPod: teste a guarda de custo e confira o Pod **EXITED**. Armazenamento cobra mesmo parado.

O que você precisa

- Um assistente que leia arquivos (Claude Code, Codex ou similar).
- Node.js na versão pedida pelo Harness e pelos servidores MCP.
- Python 3.10 ou mais novo para os scripts do kit (só biblioteca padrão; o Pillow só para recriar a imagem de teste).
- Na rota hospedada: conta e crédito de API. Na RunPod: conta e crédito. Assinatura de chat não inclui nada disso.

O que tem aqui

Prefere ler no papel? O kit inteiro está em **guia-rapido.pdf** (os Markdown continuam sendo a versão editável).

Pasta / arquivo	Para quê
prompts/	9 prompts em português, cada um com o bloco FRONTEIRAS
guias/	escolha da rota, as 3 rotas, ferramentas, segurança do modelo, custo, problemas comuns
config/	exemplos sem credencial: campos do provedor, MCP, modelos de exemplo, nomes de variáveis
scripts/doutor.py	confere o que está instalado, sem instalar nada
scripts/checa_endpoint.py	testa texto em 2 turnos, função inofensiva e imagem sintética
scripts/laboratorio.py + cheragem_segura.py	laboratório falso em 127.0.0.1 e a checagem com escopo
scripts/guarda_ociosa.py	guarda de tempo ocioso para o SEU Pod da RunPod (simulação por padrão)
exemplos/	escopo de exemplo, relatório de achados, template Nuclei passivo, imagem de teste
testes/	testes offline dos scripts
verificacao/	o que foi testado e o checklist para a sua instalação

O kit **não** traz pesos de modelo, não aluga GPU e não chama nenhum serviço sozinho. Modelos e preços são exemplos conferidos em 26/10/2026 e mudam.

Uso defensivo — as regras do kit

Este kit serve para aprender, achar fraquezas em sistemas **seus** ou que você tem **permissão por escrito** para testar, e confirmar correções. Ele não dá permissão para testar nada.

1. Escopo antes de tudo

Registre: dono do sistema, como a permissão foi dada (e onde está escrita), o host ou app exato, as ações permitidas, a janela de horário, o limite de pedidos e as condições de parada. Faltou algo? Resolva antes de mandar o primeiro pedido.

URL pública, login que funciona ou chave de API **não são permissão**. Permissão para um app não cobre serviços ligados a ele, outras contas, hosts vizinhos nem a infraestrutura do provedor.

Modelo de escopo: **exemplos/escopo-autorizado.exemplo.json**. Do jeito que vem, o kit recusa — de propósito.

2. Laboratório sintético primeiro

Comece no laboratório do kit (`scripts/laboratorio.py` , só em 127.0.0.1, dados inventados) ou num ambiente de homologação com dados falsos. Primeiro as checagens que não mudam nada.

3. Prova proporcional

Junte só a evidência necessária para confirmar o achado. Não baixe banco de dados, não exiba credencial real, não mexa em conta de outra pessoa, não guarde dado pessoal à toa. Mascare antes de compartilhar.

Pare quando o próximo passo sair do combinado, afetar a disponibilidade, expor registro sensível ou exigir mudança destrutiva. Anote o que aconteceu e peça autorização específica. Nada de persistência, disfarce ou acesso além do escopo.

4. Proteja o seu próprio ambiente

- Credencial em arquivo privado (`chmod 600`) ou cofre de segredos. Nunca em prompt, commit ou captura de tela.
- Credenciais de teste, limitadas, revogadas no fim.
- Modelo baixado roda **primeiro sem ferramentas**; depois só com o acesso a arquivos, rede e ferramentas que a tarefa pede.
- **Prompt não é cerca**. O isolamento é imposto pelo sistema operacional, pelo contêiner e pelas permissões.
- Endpoint local preso em `127.0.0.1` . Endpoint na nuvem **sempre com autenticação**.
- Antes de subir GPU, revise preço por hora, armazenamento e como parar. No fim, **pare o Pod e confira EXITED**. Armazenamento ainda cobra.

5. Resultado de IA é para conferir

A IA inventa achados e comandos errados. Scanner deixa passar falha e acusa o que não existe. Confira a evidência, reproduza com segurança, aplique uma correção revisada e reteste. Nada neste kit é certificação, garantia ou substituto de uma avaliação de segurança independente.

6. Ao publicar

Não publique credencial de terceiros, dado privado, log bruto de sessão nem alvo real sem permissão. Confira licenças antes de redistribuir código, imagem ou arquivo de modelo.

Material educativo, fornecido "como está", sem garantia. Você responde por permissões, cópias de segurança, credenciais, dados pessoais, termos de terceiros e cobranças.

1. Escolha a sua rota

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

Um modelo "sem censura" (abliterated) é um modelo aberto do qual alguém removeu, por técnica, a tendência de recusar pedidos. Para segurança defensiva isso ajuda: ele explica um cabeçalho fraco ou analisa um log sem sermão. Também quer dizer que **a responsabilidade passa a ser sua**: as cercas agora são o escopo, o laboratório e as permissões que você dá.

Rota	Combina com	Você paga	Sua responsabilidade
API hospedada	quem quer começar rápido e não tem GPU boa	uso da API (assinatura de chat não inclui API)	confiar no provedor, guardar a chave, limite de gasto
GPU alugada na RunPod	quem quer escolher modelo e runtime além do que o computador aguenta	hora de GPU + armazenamento	autenticação, atualização, parar o Pod
Local no LM Studio	quem tem máquina com memória sobrando	luz, disco e tempo; sem conta de GPU	downloads, versões do runtime, permissões locais

Ordem de teste, em qualquer rota: texto → uma função inofensiva → imagem (só se precisar). Uma camada por vez. Um modelo pode responder "oi" e falhar justo na ferramenta ou na imagem que você precisa.

Já tem uma rota funcionando? Mantenha como reserva enquanto testa outra. Não troque o que funciona só para ficar igual ao guia.

Próximo: **rota hospedada · rota RunPod · rota local**

2. Rota API hospedada

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

Antes

Rode o prompt **01** (Harness pronto) e depois o **02**. Você precisa de uma conta num provedor que aceite uso defensivo e sirva um modelo com protocolo compatível com o Harness. Pode haver verificação de identidade e cobrança separada da sua assinatura de chat. Exemplos de provedor: Abliteration e OrcaRouter. Termos, modelos e preços mudam: confira no site de cada um (conferido em 26/10/2026).

Conectar

1. Na documentação oficial do provedor, copie a **URL base** e o **protocolo** (normalmente "OpenAI compatible").
2. Crie uma chave só para teste e, se o provedor deixar, um **limite de gasto**.
3. No Harness: **Settings > Models > Add model provider > Custom model API**. Use `kit/config/campos-do-provedor.json` como folha de preenchimento.
4. Busque a lista de modelos ou digite o ID exato. Nome na tela e ID podem ser diferentes.
5. Abra **uma sessão nova** na pasta de teste. Sessões antigas podem continuar com o modelo anterior.

Verificar

Guarde a chave num arquivo privado e aponte a variável para ele:

```
mkdir -p ~/.segredos && chmod 700 ~/.segredos
nano ~/.segredos/chave-modelo      # cole a chave, salve; uma linha só
chmod 600 ~/.segredos/chave-modelo
export INEMA_CHAVE_MODELO_ARQ=~/.segredos/chave-modelo
python3 kit/scripts/checa_endpoint.py --base-url https://SEU-PROVEDOR/v1 --modelo SEU-ID --funcao
```

O checador manda só textos inventados e gasta um pouco de uso da API. Para imagem, acrescente `--imagem kit/exemplos/teste-visao.png`. Depois repita a função inofensiva dentro do próprio Harness.

Desfazer

Feche a sessão de teste, desative só o provedor que você criou e revogue a chave de teste. Olhe o painel de uso do provedor. Desinstalar o Harness **não** cancela cobrança nem revoga chave.

Documentação oficial: [provedores no Harness](#) · [Abliteration](#) · [OrcaRouter](#)

3. Rota GPU alugada na RunPod

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

Antes de alugar

Rode o prompt **04**. Peça ao assistente para separar cinco coisas: pesos do modelo, projetor de visão, runtime de inferência, tamanho de contexto e memória extra. O maior modelo que cabe no bolso nem sempre é o que chama ferramenta direito.

Aprove **por escrito** um orçamento com: GPU exata, quantidade, região, preço por hora, tamanho e preço do armazenamento, tempo máximo e total. Orçamento não garante que a GPU vai estar disponível depois. Não dê carta branca para o assistente "tentar uma GPU maior" sozinho.

Exemplo usado no kit (não é recomendação): Huihui Qwen3.5 9B abliterated, GGUF Q6_K, com o projetor de visão do mesmo publicador. Detalhes em `modelos-de-exemplo.json`.

Subir o que foi aprovado

Rode o prompt **05** com a conta da RunPod ligada pelo plugin ou CLI oficial. Use um template de inferência confiável ou um runtime com versão fixa. Modelo, projetor e configuração de inicialização no armazenamento persistente. O kit **não** aluga GPU sozinho.

Exponha só um endpoint de modelo **com autenticação**; nunca terminal ou painel de administração abertos. A chave de controle da RunPod é uma; a chave do modelo é outra.

Testes de aceitação

- [] Texto em duas rodadas de conversa.
- [] Função inofensiva com ida e volta (`checa_endpoint.py --funcao`).
- [] Imagem sintética, se a imagem fez parte da escolha.
- [] Requisição **sem chave** é recusada.
- [] Parar → conferir **EXITED** no painel → ligar de novo → modelo e configuração continuam lá → repetir o teste.

Fim da sessão

Siga **custo-runpod**. Pare **este** Pod. O armazenamento continua cobrando com o Pod parado. Arquivo no disco do contêiner pode sumir ao parar; volume do Pod some ao apagar o Pod. Só apague depois de decidir o que guardar e fazer cópia.

Documentação oficial: [Pods](#) · [preços](#) · [armazenamento](#)

4. Rota local no LM Studio

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

Escolher e carregar

Rode o prompt **07**. Deixe folga para o sistema, os outros programas, o contexto e o projetor de visão. Olhe a memória **disponível**, não só a total. Leia o cartão do modelo e a licença. Número de downloads não é certificado de segurança.

Exemplo usado no kit: Huihui Qwen3.6 35B-A3B abilitated, GGUF Q4_K (~21,7 GB, mais ~0,9 GB do projetor de visão). Não serve para todo computador. Conferido em 26/10/2026; confira de novo antes de baixar.

Rode o prompt **08**. No LM Studio, carregue o modelo e ligue o servidor na aba Developer, com o acesso pela rede **desligado**. Pela linha de comando:

```
lms ls
lms load SEU-MODELO --context-length 4096 --identifier defensivo-local --ttl 600
lms server start --bind 127.0.0.1 --port 1234
```

O contexto 4096 é um começo conservador. Se não carregar, use modelo ou contexto menor, ou feche programas. Não force além da memória livre.

Sobre imagem

No resultado observado em 26/10/2026 com esse Q4_K e o runtime daquela instalação, **texto e função passaram e a imagem falhou**. Não é prova de visão funcionando. Se precisar de captura de tela, confirme o projetor certo e o suporte do runtime, e só siga depois que o teste de imagem passar.

Ligar ao Harness

Settings > Models > Add model provider > Custom model API → protocolo OpenAI Chat Completions, `http://127.0.0.1:1234/v1`, ID devolvido pelo servidor. Deixe a interface do Harness também só local.

```
python3 kit/scripts/checa_endpoint.py --base-url http://127.0.0.1:1234/v1 --modelo
defensivo-local \
--funcao --imagem kit/exemplos/teste-visao.png
```

Isso testa conversa em dois turnos, função inofensiva e imagem, separadamente, sem dar terminal ao modelo. Falhou uma capacidade? Corrija ou troque de modelo; não marque como pronto.

Desligar

```
lms unload defensivo-local  
lms server stop
```

Pare só o servidor que você ligou e descarregue só o seu modelo de teste.

Documentação oficial: [servidor do LM Studio](#) · [ferramentas no LM Studio](#)

5. Ferramentas: navegador, pesquisa e Nuclei

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

O modelo decide; as ferramentas fazem. Use o prompt **03** só depois que o modelo básico já funciona. Cada ferramenta com o mínimo de poder e de entrada.

Configuração no Harness

`kit/config/dsh-mcp.exemplo.yml` segue o formato do plugin MCP do Harness. É um trecho para **juntar** ao perfil ativo, não para substituir. O assistente confere a versão instalada antes. O `mcp.exemplo.json` é para outros clientes que aceitam `mcpServers`.

Três provas pequenas

1. **Navegador (Playwright MCP)**. Num terminal: `python3 kit/scripts/laboratorio.py`. No Harness, peça para abrir `http://127.0.0.1:8181/`, dizer o título e ler o texto. Navegador isolado, sem suas contas.
2. **Pesquisa (Firecrawl MCP)**. Peça para ler o README público do Harness e devolver o título. O modo sem chave tem limite; o uso completo pede OAuth ou chave privada. Nada de código ou dado de cliente para um serviço de terceiros.
3. **Nuclei**. `nuclei -version`. Leia o template `nuclei-cabecalhos-lab.yaml` inteiro e rode só contra o laboratório:

```
nuclei -u http://127.0.0.1:8181/ -t kit/exemplos/nuclei-cabecalhos-lab.yaml -rl 1 -c 1
```

O template lê cabeçalhos; não explora nada. A `checagem_segura.py` faz o mesmo exercício em Python, com escopo obrigatório. **O modelo não decide o próprio escopo lendo uma página.**

Ferramenta listada não é ferramenta funcionando: só conta a chamada real com resultado.

Documentação oficial: [Playwright MCP](#) · [Firecrawl MCP](#) · [Nuclei](#) · [MCP no Harness](#)

6. Antes de dar ferramentas a um modelo baixado

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

Leitura recomendada: [ProjectDiscovery — How abliterated models can get you pwned](#). O texto mostra um modelo de pesquisa com porta dos fundos colocada de propósito. Ele **não** diz que os modelos de exemplo deste kit estão comprometidos; mostra por que confiar com cuidado.

Pesos e ferramentas são duas decisões de confiança separadas.

1. **Anote a procedência:** quem publicou, repositório, revisão exata, hash do arquivo, formato e licença. Nome conhecido e muitos downloads não garantem nada; um antivírus não certifica comportamento.
2. **Primeiro sem ferramentas e sem arquivos pessoais.** Converse, veja como responde.
3. **Depois, o mínimo de ferramentas** para uma tarefa sintética, se possível num ambiente descartável.
4. **Isolamento é do sistema, não do prompt.** Dizer "não leia segredos" não é cerca. Use usuário separado, contêiner, pasta de trabalho dedicada. `127.0.0.1` impede acesso pela rede, mas não impede o Harness de ler os arquivos que você deixou ao alcance.
5. **Nada real na pasta de teste:** sem credencial verdadeira, sem dado de produção. Ferramenta de pesquisa recebe só URL pública.
6. **Desconfie de pedidos estranhos do modelo:** baixar e rodar script remoto, mostrar chave, desligar proteção, aumentar escopo. Pare e investigue. Não teste payload malicioso na sua máquina de trabalho.
7. **Mudou algo, teste de novo:** pesos, projetor, template de chat, runtime ou permissões.

Documentação oficial: [SAFETY.md do Harness](#) · [segurança no Hugging Face](#)

7. Custo na RunPod: parar de verdade

Uso defensivo: só o seu Pod aprovado. Credenciais fora do chat.

Parar o app não é parar a GPU. Confira sempre o Pod **EXITED** no painel. O armazenamento persistente continua cobrando com o Pod parado.

Primeiro, o limite manual

Anote ID e nome exatos do Pod, preço por hora aprovado, custo do armazenamento e a hora de parar. Ligue os alertas de orçamento da conta, se houver. No fim, pare **esse** Pod no painel e leia o status. Só apague depois de decidir o que guardar.

Guarda ociosa (opcional)

`guarda_ociosa.py` é um proxy local: repassa os pedidos ao modelo e mede o tempo sem uso. Escuta só em `127.0.0.1:8282`, exige um token local, não registra pedidos nem chaves e começa em **simulação**. Só para um Pod se você mandar, com ID e nome exatos. Nunca cria, liga ou apaga Pod.

É ajuda, não garantia: não vê clientes que falam direto com o Pod; se o computador dormir, o processo cair ou a rede falhar, ela não para a GPU. Use junto com a parada manual e o alerta do provedor.

```
mkdir -p ~/.segredos && chmod 700 ~/.segredos
python3 -c "import secrets;print(secrets.token_urlsafe(32))" > ~/.segredos/token-guarda
chmod 600 ~/.segredos/token-guarda
export INEMA_TOKEN_LOCAL_ARQ=~/.segredos/token-guarda
export INEMA_CHAVE_MODELO_ARQ=~/.segredos/chave-modelo      # a chave do endpoint do modelo
(chmod 600)
python3 kit/scripts/guarda_ociosa.py --upstream https://SEU-ENDPOINT/v1 --ocioso 900
```

1. No Harness, aponte o provedor para `http://127.0.0.1:8282/v1` e use o **token local** (não a chave da RunPod). ID do modelo continua o original. **Todo** pedido passa por aqui.
2. Mande um pedido, espere, mande outro: o relógio zera. Um pedido em andamento ou em streaming impede a parada. Depois do tempo ocioso o terminal mostra `{"guarda_ociosa": "simulacao_pararia"}`.
3. Para ligar a parada real, guarde a chave de controle da RunPod em arquivo privado e reinicie:

```
export INEMA_CHAVE_RUNPOD_ARQ=~/.segredos/chave-runpod      # chmod 600
python3 kit/scripts/guarda_ociosa.py --upstream https://SEU-ENDPOINT/v1 --ocioso 900 \
  --parar-de-verdade --pod-id SEU_POD_ID --pod-nome "NOME-EXATO-DO-POD"
```

A guarda lê o Pod, confere ID e nome, pede a parada e espera `desiredStatus: EXITED`. Se não confirmar, mostra `parada_falhou_ou_nao_confirmada` e tenta de novo só depois de outro período ocioso. Na primeira vez, confira você mesmo no painel: pedido de parada não é prova.

Ligar de novo e remover

Ligue o Pod pelo painel, confirme endpoint e autenticação, e só então reinicie a guarda. Ela não liga GPU paga sozinha, de propósito. Para remover: pare o processo, volte o provedor do Harness ao endpoint original e revogue a chave de controle.

Pedidos de lista de modelos também contam como uso: um cliente que consulta a lista de tempos em tempos pode manter o Pod acordado. A guarda foi feita para um usuário num Pod dedicado.

Documentação oficial: [parar Pod](#) · [status do Pod](#) · [armazenamento](#)

8. Problemas comuns: conserte a menor camada

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

Sintoma	Confira primeiro	Próximo passo
Conexão recusada	o servidor está ligado na porta certa?	ver o status do servidor; manter 127.0.0.1 igual em todo lugar
Erro 401 ou 403	chave certa e conta com permissão?	digitar a chave de novo no lugar privado; nunca colar no relatório
Lista de modelos vazia	o endpoint tem <code>/v1/models</code> ?	ler a documentação e digitar o ID exato
Responde "oi" mas a função falha	modelo e runtime suportam chamada de função?	rodar <code>checa_endpoint.py --funcao</code> ; olhar o template de chat
Ignora a imagem	visão ligada e projetor certo?	rodar só o teste <code>--imagem</code>
Modelo não carrega	memória livre e runtime compatível?	contexto ou modelo menor; não forçar sem memória
Harness continua com o modelo antigo	sessões antigas guardam o modelo	abrir sessão nova e conferir
MCP aparece mas não faz nada	uma chamada real deu certo?	olhar a inicialização do servidor MCP e fazer um teste pequeno
A nuvem continua cobrando	parou o Pod ou só o app?	conferir EXITED; olhar o armazenamento à parte
Guarda ociosa não fez nada	todo pedido passou por ela? ela estava rodando?	manter parada manual e alerta de orçamento
<code>checagem_segura.py</code> recusou	é o laboratório em 127.0.0.1 e o escopo está valendo?	ler o motivo no JSON; a recusa é proposital

Deu errado? Desfaça só o que você mudou. Guarde os perfis que funcionam. Nunca resolva conexão abrindo um endpoint sem autenticação, nem colando chave numa issue pública.

Configure com o seu assistente

Uso defensivo: só sistemas seus ou com permissão por escrito. Credenciais fora do chat.

Entregue ao seu assistente (Claude Code, Codex ou similar) a pasta deste repositório e cole o pedido abaixo. Se o assistente não abre links, anexe os arquivos citados.

Me ajude a usar o kit "IA sem censura, do jeito certo" para estudo de segurança defensiva autorizado.

Leia primeiro: `kit/COMECE-AQUI.md`, `kit/USO-DEFENSIVO.md`, `kit/verificacao/STATUS.md` e `kit/guias/1-escolha-da-rota.md`.

Pergunte: qual rota eu quero, meu sistema e memória livre, se preciso de imagem e qual é o meu orçamento.

Reaproveite o que já está instalado. Nunca me peça para colar chave neste chat.

Use os prompts numerados da rota escolhida (`kit/prompts/`). Antes de mudar configuração, confira a documentação oficial e a versão instalada e me mostre os arquivos e as mudanças. Não sobrescreva provedores nem MCPs que já existem.

Não crie nada pago sem meu OK explícito para a GPU exata, o preço por hora, o armazenamento e o tempo máximo.

Prove de verdade: uma resposta, uma conversa de dois turnos e uma função inofensiva (`python3 kit/scripts/checa_endpoint.py ... --funcao`). Imagem só se eu pedir, em teste separado.

Falhou? Diga que falhou; item na lista não é item funcionando.

Testes de segurança só no laboratório sintético (`kit/scripts/laboratorio.py` + `kit/scripts/cheragem_segura.py`).

Para qualquer outro alvo: minha posse ou permissão escrita, escopo exato, limites e condições de parada.

Faltou algo, pare.

No fim, me entregue: modelo e versão, endpoint, onde ficaram as credenciais (sem os valores), resultado de cada teste e como ligar, parar e desinstalar.

Depois de colar, acompanhe: o assistente deve **mostrar** antes de **fazer**, e parar antes de gastar.

Status da verificação — 26/10/2026

Testes com limite claro. Não é certificação de segurança.

Camada	Status	Quem rodou
Testes offline dos scripts (<code>kit/testes/</code>)	25 testes passando: escopo negado, alvo remoto recusado, escopo vencido, janela sem fuso, limite de pedidos, achado sintético + correção, servidor que não é o laboratório, URL e permissão de chave, relógio ocioso (simulação, pedido em andamento, falha de parada), parada de Pod contra API falsa (nome errado, EXITED confirmado, sem EXITED), proxy (sem token, origem de navegador, rota estranha, só loopback), checador de endpoint contra modelo falso	INEMA, em 127.0.0.1, sem internet
Laboratório + checagem segura pela linha de comando	achados aparecem; com <code>--corrigido</code> somem; escopo de exemplo e alvo remoto recusados	INEMA
Modelo local Huihui Qwen3.6 35B-A3B abilitated Q4_K no LM Studio	resultado observado em 26/10/2026 numa instalação existente: texto em 2 turnos ✓, função inofensiva ✓, imagem ✗ nesse runtime. Não foi repetido nesta versão do kit	—
Modelo na nuvem Huihui Qwen3.5 9B abilitated Q6_K	não implantado: nenhuma GPU paga foi alugada para este kit	—
Provedor hospedado pago	não testado: depende da conta e da chave de quem usa	—
Parada real de Pod na RunPod	só testada contra API falsa; a parada real exige o seu Pod e a sua chave	—
Template Nuclei passivo	escrito para o laboratório; não executado pelo INEMA (Nuclei não instalado na máquina de teste)	—
DeepSeek Harness, Playwright MCP, Firecrawl MCP	configuração de exemplo; não instalados pelo INEMA nesta versão	—
Windows / macOS / Linux	instruções incluídas; sem prova de instalação nova em cada sistema	—

Use o **checklist** na sua instalação. Link de documentação não é teste ao vivo.

Checklist da sua instalação

Uso defensivo. Cada item contra o SEU laboratório ou o SEU endpoint aprovado.

- [] Anotei sistema, versões de Python / Node / Harness / runtime e o ID exato do modelo.
- [] `python3 kit/scripts/doutor.py`
- [] `python3 -m unittest discover -s kit/testes -v` (testes offline, sem internet)
- [] Terminal 1: `python3 kit/scripts/laboratorio.py` · Terminal 2: `python3 kit/scripts/checagem_segura.py --lab-local-autorizado` → aparecem 2 achados
- [] Parei o laboratório, liguei com `--corrigido`, repeti a checagem → "achados": []
- [] `checa_endpoint.py` com o ID exato: texto, função e imagem são **resultados separados**
- [] Repeti uma função inofensiva dentro do Harness
- [] Nuvem: pedido **sem chave** é recusado
- [] Guarda ociosa em simulação: pedido zera o relógio; pedido em andamento impede parada; aparece `simulacao_pararia`
- [] Primeira parada real conferida no painel (EXITED); liguei de novo e o modelo continua lá
- [] Anotei o que **não** testei. Teste com API falsa não prova conta real, GPU paga nem integração no app.

Os 9 prompts

Copie na ordem da sua rota. Cada um traz o bloco de segurança; não tire.

01-instalar-harness

PROMPT 01 – Instalar (ou reaproveitar) o DeepSeek Harness

Quero usar o DeepSeek Harness (<https://github.com/deepseek-ai/deepseek-harness>) para estudo de segurança defensiva. Primeiro olhe o que já existe: meu sistema operacional, a versão do Node.js e se o Harness já está instalado. Se já estiver funcionando, reaproveite; não atualize nem sobrescreva configurações sem me perguntar. Leia o README e o SAFETY.md do projeto. Crie uma pasta de trabalho só para testes, longe dos meus arquivos pessoais, e me mostre o plano de instalação antes de executar. Suba a interface só em 127.0.0.1, adicione apenas essa pasta de teste e confirme que a tela abre. No fim, me diga: versão instalada, endereço local, como ligar e desligar e quais arquivos foram alterados.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (chmod 600).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

02-conectar-provedor-hospedado

PROMPT 02 – Conectar um modelo hospedado (rota API)

Quero ligar um modelo hospedado ao meu DeepSeek Harness. Eu escolho o provedor e o orçamento. Se eu ainda não escolhi, compare a documentação oficial atual de dois provedores de modelos abliterated (exemplos: Abliteration e OrcaRouter) sem supor que algum é livre de risco: exigências de conta, termos de uso e preço por token de hoje. No Harness, em Settings > Models, adicione um provedor de API personalizado com o protocolo e a URL base da documentação; busque a lista de modelos e confirme comigo o ID exato. Eu mesmo digito a chave no app ou num arquivo privado. Depois verifique: duas rodadas de conversa e uma chamada de função inofensiva; imagem só se eu precisar, num teste separado. Não mexa nos provedores que já existem e me diga o que passou e o que falhou.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (chmod 600).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

03-adicionar-ferramentas

PROMPT 03 – Dar ferramentas ao modelo (navegador, pesquisa e Nuclei)

Com o modelo já respondendo, quero adicionar três ferramentas: Playwright MCP (navegador), Firecrawl MCP (pesquisa em páginas públicas) e o Nuclei pelo terminal. Antes, leia o formato de MCP da versão instalada do Harness e compare com `kit/config/dsh-mcp.exemplo.yml`; não cole um JSON genérico 'mcpServers' no perfil do Harness. Use um perfil de navegador isolado, sem minhas contas logadas. Para o Firecrawl, use o modo sem chave com limite para o primeiro teste, ou guarde a chave num cofre privado. Prove cada ferramenta com uma chamada real: Playwright abrindo o laboratório sintético (`python3 kit/scripts/laboratorio.py`, em `http://127.0.0.1:8181/`), Firecrawl lendo uma página pública de documentação, e Nuclei com '`nuclei -version`' mais o template passivo `kit/exemplos/nuclei-cabecalhos-lab.yaml` SÓ contra o laboratório local. Nada de conjuntos grandes de templates. Ferramenta aparecer na lista não é ferramenta funcionando. Se precisar desfazer, remova só o que você criou.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (`chmod 600`).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

04-pesquisar-modelos-nuvem

PROMPT 04 – Escolher um modelo para a nuvem (só pesquisa, sem gastar)

Pesquise três modelos absterger no Hugging Face que caibam numa GPU pequena da RunPod e falem com o DeepSeek Harness por uma API compatível com a OpenAI. Preciso de texto, chamada de função e, se possível, imagem. Compare: quem publicou, repositório, revisão e arquivo exatos, suporte no runtime, memória necessária (incluindo contexto e cache), projetor de visão, custo por hora da GPU e do armazenamento, disponibilidade, licença e dúvidas de procedência. Use kit/config/modelos-de-exemplo.json só como ponto de partida datado, não como preço atual. Pergunte meu teto por hora, o orçamento total e quanto tempo vou usar. Velocidade estimada é estimativa: diga isso. Recomende um e explique por que o segundo perdeu. Não crie nenhum recurso pago.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (chmod 600).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

05-subir-modelo-nuvem

PROMPT 05 – Subir na RunPod o modelo que eu aprovei

Suba na RunPod apenas o modelo, a GPU e o runtime que aprovei depois do prompt 04. Antes de criar qualquer coisa, reconfirme comigo: preço por hora, custo do armazenamento, região e o tempo máximo de uso. Não mexa em Pods que já existem. Fixe a revisão do modelo e o projetor de visão compatível; guarde modelo e configuração de inicialização no armazenamento persistente. Crie um endpoint compatível com OpenAI COM autenticação e prove que uma requisição sem chave é recusada. A chave de controle da RunPod é diferente da chave do modelo e as duas ficam fora do chat. Conecte o ID exato no Harness. Teste: duas rodadas de conversa, uma função inofensiva e a imagem sintética kit/exemplos/teste-visao.png. Depois siga kit/guias/custo-runpod.md: pare, confira EXITED, ligue de novo e confirme que os arquivos continuam lá. Me entregue endpoint, versões, resultado de cada teste, o que continua cobrando e como parar ou apagar ESTE Pod. Se alguma capacidade prometida falhar, não diga que terminou.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (chmod 600).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

O6-guarda-ociosa

PROMPT 06 – Ligar e provar a guarda de tempo ocioso

Para o Pod que eu aprovar, configure a guarda local `kit/scripts/guarda_ociosa.py` seguindo `kit/guias/custo-runpod.md`. Comece em simulação. Faça TODO pedido ao modelo passar por ela (`http://127.0.0.1:8282/v1`). Prove três coisas: um pedido zera o relógio; um pedido em andamento ou em streaming impede a parada; depois do tempo ocioso aparece

'`simulacao_pararia`'. Só depois do meu OK, ligue a parada real com `--parar-de-verdade`, `--pod-id` e `--pod-nome` exatos e a chave de controle num arquivo privado. Confira ID e nome antes de parar e confirme EXITED depois; ligue de novo num teste separado. Explique que a guarda não vê clientes que falam direto com o Pod, que computador dormindo ou processo caído não param a GPU, e que o armazenamento continua cobrando. Mantenha a parada manual e o alerta de orçamento do provedor. Nunca crie nem apague Pod, e nunca pare um Pod que não seja este.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (`chmod 600`).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

07-pesquisar-modelos-locais

PROMPT 07 – Escolher um modelo para o meu computador

Veja meu sistema, CPU, GPU, disco livre e memória disponível, sem abrir meus arquivos pessoais. Encontre três modelos abilitados no Hugging Face que caibam com folga para contexto, cache e projetor de visão (se eu precisar de imagem). Preciso de texto e chamada de função. Compare: quem publicou, revisão, quantização, licença, procedência, suporte no runtime e tamanho do download. Se eu já tiver um modelo compatível baixado, prefira ele. Tokens por segundo são estimativas não testadas: deixe isso claro. Lembre que popularidade e antivírus não provam que um modelo é seguro. Recomende um, explique por que os outros perderam e pergunte antes de qualquer download grande.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (chmod 600).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

08-conectar-modelo-local

PROMPT 08 – Ligar o modelo local do LM Studio ao Harness

Use o modelo local que aprovei no LM Studio e ligue ao DeepSeek Harness. Confira as versões do LM Studio e do runtime. Reaproveite o que já foi baixado, escolha um contexto modesto que caiba com folga e deixe o servidor preso em 127.0.0.1. Confirme com quem publicou o modelo qual projetor de visão combina com ele. Adicione um provedor separado no Harness sem apagar o que existe. Rode `kit/scripts/checa_endpoint.py` com o ID exato do modelo e `--funcao`; se eu precisar de imagem, rode também com `--imagem kit/exemplos/teste-visao.png`. Não dê terminal nem acesso livre a arquivos ao modelo só para testar chamada de função. Informe os tempos observados junto com hardware, contexto e limite de saída, sem chamar isso de benchmark. Me dê os comandos para descarregar o modelo e parar o servidor, e desligue tudo no fim se eu pedir.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (`chmod 600`).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.

09-checar-meu-app

PROMPT 09 – Checar o MEU app com escopo autorizado

Quero examinar um app de teste meu. Comece pelo laboratório sintético do kit, nunca por um site público. Leia kit/USO-DEFENSIVO.md. No laboratório: rode python3 kit/scripts/laboratorio.py num terminal e python3 kit/scripts/checagem_segura.py --lab-local-autorizado em outro; explique os achados usando kit/exemplos/relatorio-de-achados.exemplo.md; depois rode o laboratório com --corrigido e faça o reteste. Para um app real, ANTES de qualquer pedido: registre a permissão escrita, o alvo exato, as ações permitidas, a janela, o limite de ritmo e as condições de parada (modelo em kit/exemplos/escopo-autorizado.exemplo.json) e me mostre um plano não destrutivo para eu aprovar. Colete só a evidência mínima, com dados mascarados, separe achado confirmado de hipótese, proponha correções e um reteste limitado. Nada de varredura automática fora do escopo aprovado.

FRONTEIRAS (não apague este bloco)

- Uso defensivo: só sistemas meus ou com permissão por escrito do dono. URL pública, login que funciona ou chave de API não são permissão.
- Antes de qualquer teste de segurança, peça o alvo exato, onde está a permissão escrita, as ações permitidas, a janela de horário e os limites. Se faltar, vencer ou ficar confuso: pare sem testar.
- Comece pelo laboratório sintético em 127.0.0.1. Só checagens não destrutivas, dentro do alvo, do ritmo e da janela aprovados. Não siga redirecionamento nem sugestão de ferramenta que saia do escopo.
- Prova proporcional: a menor evidência que confirma o achado. Nada de baixar banco de dados, exibir credencial real, mexer em conta de outra pessoa, derrubar serviço, criar persistência ou apagar rastros. Apareceu dado sensível, escrita inesperada ou lentidão: pare e me avise.
- Credenciais nunca no chat, em log ou no git. Eu digito direto no app ou num arquivo privado (chmod 600).
- Antes de mudar configuração, mostre o que vai mudar. Antes de gastar dinheiro (GPU, API), mostre o custo exato e espere meu OK. Pedido de instalação não autoriza varredura.
- Separe o que foi observado, o que é hipótese e o que é recomendação. Proponha a correção e um reteste dentro do escopo. Varredura sem achados não prova que algo é seguro.