MÓDULO 02

Segurança *Blindando Seu Agente*

Telegram allowlist, firewall UFW, fail2ban e credenciais seguras

DURAÇÃO	FORMATO	KIT
15 min	**Live coding**	**2 arquivos**

Por Que Agora?

Real: 1.015+ Tentativas de Brute Force em 24h

Servidores expostos na internet recebem MILHARES de tentativas de invasão por dia. Se você não blindar AGORA, alguém vai entrar.

Segurança não é opcional. É literalmente a diferença entre "meu agente funciona" e "hackearam meu servidor e roubaram minhas API keys".

O Que Vamos Blindar

1. **Telegram Allowlist (dmPolicy)**
   Garantir que APENAS VOCÊ pode comandar o bot. Se estiver "open", qualquer um pode usar.

2. **Firewall UFW**
   Bloquear TODAS as portas exceto SSH. Zero exposição desnecessária.

3. **Fail2ban**
   Detectar e banir IPs que tentam brute force no SSH (5 tentativas → ban 1h).

4. **Credenciais Seguras**
   NUNCA hardcodar API keys. Use 1Password CLI ou variáveis de ambiente.

5. **Portas Internas**
   Gateway deve escutar em 127.0.0.1 (localhost), não 0.0.0.0 (mundo inteiro). Use Cloudflare Tunnel se precisar acesso externo.

Passo 1: Configurar Telegram Allowlist

Este é o MAIS IMPORTANTE. Se o dmPolicy estiver "open", qualquer pessoa que encontrar seu bot pode comandá-lo.

```bash
# Ver config atual
cat ~/.openclaw/openclaw.json | grep dmPolicy
```

Deve estar assim:

```json
{
  "telegram": {
    "dmPolicy": "allowlist",
    "dmAllowlist": ["123456789"]
  }
}
```

> ⚠ **Se Estiver "open"**
>
> Pare TUDO e mude pra "allowlist" AGORA. Todo segundo que passa, alguém pode estar usando seu agente.

Como pegar seu Telegram ID?

```text
# Envie no chat com seu bot:
/start

# O agente responde com seu ID, tipo:
Seu Telegram ID: 123456789
```

Passo 2: Configurar Firewall UFW

UFW (Uncomplicated Firewall) é simples e poderoso. Vamos bloquear TUDO exceto SSH.

```bash
# Definir política padrão: BLOQUEAR tudo
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Permitir SSH (senão você se tranca pra fora!)
sudo ufw allow ssh

# Ativar firewall
sudo ufw --force enable
```

```
# Verificar status
sudo ufw status verbose
```

Resultado

Agora APENAS a porta SSH (22) está acessível. Tudo mais? Bloqueado.

Passo 3: Instalar e Configurar Fail2ban

Fail2ban monitora logs e bane IPs que tentam brute force. Configuração simples: 5 tentativas erradas de SSH → ban por 1 hora.

```bash
# Instalar fail2ban
sudo apt update
sudo apt install fail2ban -y

# Criar configuração local
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local

# Editar config (ou deixar padrão)
sudo nano /etc/fail2ban/jail.local
```

Configure a seção `[sshd]`:

```ini
[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 5
bantime = 3600
findtime = 600
```

O que isso faz:

- **maxretry = 5:** Banir após 5 tentativas falhas
- **bantime = 3600:** Ban dura 1 hora (3600 segundos)
- **findtime = 600:** 5 tentativas em 10 minutos = ban

```bash
# Reiniciar fail2ban
sudo systemctl restart fail2ban

# Ver status
sudo fail2ban-client status sshd
```

Teste Você Mesmo

Tente fazer login com senha errada 5x de outro IP. Você vai ver o ban acontecendo em tempo real.

Passo 4: Credenciais Seguras

NUNCA hardcode API keys em arquivos de config. Use 1Password CLI ou variáveis de ambiente.

Opção A: 1Password CLI (Recomendado)

```bash
# Instalar 1Password CLI
curl -sS https://downloads.1password.com/linux/keys/1password.asc | \
sudo gpg --dearmor --output /usr/share/keyrings/1password-archive-keyring.gpg

echo "deb [arch=amd64] https://downloads.1password.com/linux/debian/amd64 stable m
sudo tee /etc/apt/sources.list.d/1password.list

sudo apt update && sudo apt install 1password-cli -y

# Fazer login
op signin
```

No código, em vez de:

```javascript
const apiKey = "sk-ant-1234567890abcdef"; // □ MAL!
```

Faça:

```javascript
const apiKey = process.env.ANTHROPIC_API_KEY; // □ BOM!
```

Opção B: Variáveis de Ambiente

```bash
# Editar .env
nano ~/.openclaw/.env

# Adicionar:
ANTHROPIC_API_KEY=sk-ant-1234567890abcdef
TELEGRAM_BOT_TOKEN=123456:ABC-defgh...
```

⚠ **CUIDADO: systemd Override**

Se o gateway roda como serviço systemd, o arquivo `.service` pode sobrescrever o `.env`. Atualize AMBOS ao trocar credenciais.

Passo 5: Portas de Aplicação

O gateway OpenClaw escuta na porta 18789 por padrão. NUNCA exponha essa porta pro mundo inteiro.

```bash
# Ver config
cat ~/.openclaw/openclaw.json | grep bindAddr
```

Deve estar assim:

```json
{
  "gateway": {
    "bindAddr": "127.0.0.1:18789"
  }
}
```

Se Estiver 0.0.0.0

Isso significa que QUALQUER PESSOA na internet pode acessar o gateway. Mude pra 127.0.0.1 IMEDIATAMENTE.

E Se Eu Precisar de Acesso Externo?

Use **Cloudflare Tunnel** (grátis, seguro, zero portas expostas).

```bash
# Instalar cloudflared
curl -L https://github.com/cloudflare/cloudflared/releases/latest/download/cloudfl
sudo dpkg -i cloudflared.deb

# Criar tunnel
cloudflared tunnel login
cloudflared tunnel create openclaw-gateway
cloudflared tunnel route dns openclaw-gateway gateway.seudominio.com
```

Resultado

Agora você acessa via `gateway.seudominio.com` — porta 18789 NUNCA é exposta diretamente.

✅ Checkpoint do Módulo 2

- [ ] dmPolicy = allowlist (só você comanda o bot)
- [ ] UFW ativo (tudo bloqueado exceto SSH)
- [ ] Fail2ban protegendo SSH (5 tentativas → ban 1h)
- [ ] Credenciais no 1Password ou .env (zero hardcode)
- [ ] Gateway em 127.0.0.1 (não 0.0.0.0)

🤖 Prompt para o Agente

Cole este prompt no chat do seu OpenClaw depois de assistir o Módulo 2:

Acabei de assistir o Módulo 2 do curso sobre segurança. Leia o PRD de security hardening que estou anexando e me guie passo a passo. **Importante:** - Antes de CADA ação, me explique O QUÊ vai fazer e POR QUÊ - Me peça confirmação antes de executar qualquer mudança no servidor - Se algo já estiver configurado, me avise e pule pro próximo passo - No final, rode um audit de segurança e me mostre o resultado **O que espero que cubra:** 1. Telegram allowlist (dmPolicy) — pra ninguém comandar meu bot 2. Firewall UFW — bloquear portas desnecessárias 3. Fail2ban — proteger contra brute force SSH 4. Credenciais — como guardar API keys de forma segura 5. Portas de aplicação — não expor nada em 0.0.0.0 Vamos blindar meu servidor?

PRÓXIMO MÓDULO

Módulo 3 — Identidade: Dando Personalidade ao Agente