Roadmap Atualizado - Projeto Imunno

Versão: 2.0
Data: 15 de Julho de 2025
Autor: Sistema Automatizado de Revisão (com validação manual pelo criador)

---

1. Visão do Projeto

O **Projeto Imunno** é um sistema de segurança inspirado na inteligência do sistema imunológico humano, voltado para servidores web. Ele foi projetado para oferecer detecção inteligente, leve e autônoma de atividades maliciosas em tempo real.

Foco principal: - Pequenas empresas, freelancers e desenvolvedores que utilizam sistemas CMS como WordPress. - Cenários onde as ferramentas de segurança tradicionais são muito pesadas, caras ou complexas.

---

2. Princípios de Engenharia

Todas as decisões técnicas seguem quatro pilares:

- **Simplicidade**: Preferência por soluções robustas e diretas.
- **Clareza**: Código, arquitetura e dados devem ser legíveis e explicáveis.
- **Manutenibilidade**: Facilmente testável, expandível e modular.
- **Complexidade Justificada**: Apenas adotada quando absolutamente necessária para robustez.

---

3. Arquitetura Geral

Componentes:

Componente	Linguagem	Função
imunno-agent	Go	Monitora arquivos e processos; envia eventos via REST.
imunno-collector	Go	Recebe eventos; aplica análise heurística + IA; coordena quarentena.
imunno-ml-service	Python	API FastAPI que executa inferências usando modelo treinado.
PostgreSQL	SQL	Armazenamento de eventos e hashes confiáveis.
Dashboard/UI	HTML/JS	Interface de alerta em tempo real via WebSocket.
Docker Compose	YAML	Orquestra todo o ecossistema localmente.

Fluxo de Dados:

1. `agent` detecta novo processo ou arquivo → envia JSON via HTTP.
2. `collector` armazena evento, analisa o conteúdo estaticamente (RegEx) e chama a IA.
3. `ml-service` responde se o arquivo é anômalo + confiança.
4. `collector` decide por quarentena, broadcast WebSocket e registro.
5. UI recebe evento e exibe em tempo real.

---

4. Fases Concluídas

✅MVP Técnico

- Criado `imunno-agent` e `imunno-collector` em Go.
- Comunicação REST com JSON padronizado.

✅Profissionalização

- Separado em módulos, com configuração via `.env`.
- Banco PostgreSQL funcional com scripts automatizados.
- Orquestração completa com Docker Compose.

✅Inteligência Nível 1: Análise Estática

- Motor heurístico com expressões regulares.
- Capaz de detectar funções como `eval`, `base64_decode`, `shell_exec`, etc.
- Sistema modular: `AnalyzeContent(content []byte)` baseado em `AnalisarConteudo(string)`.

✅Inteligência Nível 2: Análise de Comportamento

- Agente monitora e envia eventos de processo.
- Collector correlaciona eventos baseando-se no tempo e host.

✅Nível 3: Integração de IA

- Modelo Random Forest treinado via Jupyter + joblib.
- `ml-service` serve o modelo via FastAPI.
- Collector chama o endpoint `/predict` e ajusta threat score com base na IA.

✅Memória Imunológica

- Whitelist baseada em SHA256.
- Script `populate_whitelist.go` converte arquivos de hash em SQL.
- Collector ignora arquivos conhecidos como confiáveis.

✅Dashboard Funcional

- WebSocket Hub no `collector` transmite eventos.
- Interface web com visual "futurista" para exibir alertas.

---

5. Pontos de Destaque Arquitetural

- **WebSocket centralizado**: permite comunicação bidirecional com agentes e UI em tempo real.
- **Banco de dados relacional**: estrutura normalizada, com separação clara de eventos e whitelist.
- **Mecanismo de quarentena**: robusto e portável, baseado em "copiar e apagar".
- **Módulos Go desacoplados**: `database`, `hub`, `analyzer`, `config` e `ml_client` organizados.

---

6. Status Atual

O Projeto Imunno alcançou a estabilidade funcional como MVP. Todas as partes essenciais estão operacionais, com destaque para:

- Integração bem-sucedida entre linguagem Go e Python via REST.
- Estabilidade no tratamento de eventos simultâneos.
- Painel responsivo e compatível com uso prático.
- Testes bem-sucedidos com WordPress real (hashes confiáveis, plugins maliciosos).

---

7. O Que Está Por Vir (Fase 4 em diante)

(Definido, mas aguardando testes de estabilidade e dados reais para avançar)

- Correlação por linhagem de processo (não apenas temporal).
- Autoaprendizado com base em detecções bem-sucedidas.
- Geração automática de novas regras heurísticas.
- Versão com foco em produção distribuída (hospedagem remota).

Fim do Roadmap 2.0

Para feedback, melhorias ou revisões futuras, comunique via GitHub Issues.