ReconSec v2.0 — Guia Corporativo de Uso e Manutenção Autor: Ivo Matos — Analista de Segurança da Informação / Befly Data: Outubro de 2025

|  |
| --- |
| 1. Introdução |

O ReconSec é uma ferramenta corporativa desenvolvida em Go (Golang) com o propósito de auxiliar analistas de segurança da informação e profissionais de AppSec na realização de tarefas de reconhecimento, análise de código e avaliação de componentes de aplicações.

Sua proposta é centralizar os processos de Recon, SCA (Software Composition Analysis), SAST (análise estática) e Testes Dinâmicos (seguros e éticos), em uma única suíte leve, modular e extensível.

O ReconSec foi desenvolvido para ambientes corporativos e uso controlado, seguindo boas práticas de segurança e conformidade. Ele não executa ações destrutivas por padrão e requer confirmação explícita para qualquer operação ativa.

Público-alvo: - Analistas de Segurança da Informação - Profissionais de AppSec - Pesquisadores de vulnerabilidades com escopo autorizado - Times de Red Team e Pentest internos

|  |
| --- |
| 2. Instalação e Pré-Requisitos |

Requisitos mínimos: - Go 1.20 ou superior - Linux, macOS ou WSL (Windows Subsystem for Linux) - Acesso à internet para baixar dependências (go mod tidy)

Instalação:

1. Clonar o repositório: git clone https://github.com/seuusuario/reconsec.git
2. Entrar no diretório: cd reconsec-v2.0
3. Atualizar dependências: go mod tidy
4. Compilar o binário: go build -o reconsec ./cmd/reconsec
5. Validar a instalação: ./reconsec version

Instalação global (opcional): sudo cp reconsec /usr/local/bin/reconsec echo “alias recon=‘reconsec’” >> ~/.bashrc source ~/.bashrc

|  |
| --- |
| 3. Estrutura de Pastas e Organização |

A organização modular facilita a manutenção, testes e evolução do ReconSec.

cmd/ → Contém o entrypoint principal (main.go). É o ponto de inicialização da CLI.

internal/ → Contém os pacotes internos não exportáveis, usados para lógica interna. cli/ — gerenciamento de comandos e flags. runner/ — orquestra os módulos (recon, scan, sca, etc).

pkg/ → Pacotes principais da ferramenta, reutilizáveis entre módulos. recon/ — Reconhecimento passivo, coleta de URLs, JS, forms. sca/ — Análise de dependências (SCA). report/ — Geração de relatórios (JSON, TXT, HTML). scanner/ — Testes dinâmicos seguros e validações de status code. utils/ — Funções auxiliares (HTTP client, sanitização).

web/ → Contém templates e arquivos estáticos para relatórios HTML.

.github/ → Workflow CI/CD e configurações de automação (build/test).

|  |
| --- |
| 4. Comandos e Funcionalidades |

Todos os comandos seguem o padrão: reconsec [flags]

| 4.1 Comando: recon |
| --- |
| Objetivo: Executar reconhecimento passivo de aplicações web. |
| Descrição: Coleta informações públicas da aplicação, como headers, robots.txt, sitemap, links, endpoints JavaScript e formulários HTML. |
| Flags: -url → endereço do alvo (obrigatório) -depth → profundidade da coleta (padrão 1) -out → salva o resultado em JSON –txt → gera um resumo em formato texto (.txt) –html → gera relatório HTML –timeout → define timeout HTTP –rate → define taxa máxima de requisições |
| Exemplo: ./reconsec recon -url “https://testphp.vulnweb.com/” -out recon.json –txt |

## 4.2 Comando: sca

Objetivo: Analisar dependências de software (SCA).

Descrição: Examina arquivos de dependência como package.json, go.mod, requirements.txt, composer.lock, etc. e identifica versões e possíveis vulnerabilidades.

Flags: -path → caminho do código -out → saída em JSON

Exemplo: ./reconsec sca -path ./meuprojeto -out sca.json

| 4.3 Comando: scan |
| --- |
| Objetivo: Analisar código-fonte local em busca de padrões perigosos. |
| Descrição: Executa uma varredura leve (SAST-lite) procurando trechos de código com possíveis execuções de comandos, injeções SQL e uso inseguro de bibliotecas. |
| Flags: -path → caminho do código a analisar -out → saída em JSON |
| Exemplo: ./reconsec scan -path ./app -out findings.json |

## 4.4 Comando: test

Objetivo: Realizar testes dinâmicos controlados.

Descrição: Executa validações passivas (por padrão) injetando tokens inocuos e verificando respostas HTTP. O modo ativo só funciona mediante confirmação explícita do analista.

Flags: -url → endereço alvo (obrigatório) –only <xss|sqli|rce|all> → define tipos de teste –mode <passive|active> → define o modo de operação –confirm-authorized “texto” → confirmação obrigatória para modo ativo -out → arquivo de saída JSON

Exemplo (modo passivo): ./reconsec test -url “https://testphp.vulnweb.com/” -out tests.json

Exemplo (modo ativo, com autorização): ./reconsec test -url “https://exemplo.com” –mode active –confirm-authorized “Tenho autorização para realizar testes.” -out active.json

| 4.5 Comando: full |
| --- |
| Objetivo: Executar o pipeline completo (Recon + SCA + Scan + Test + Relatório). |
| Flags: -url → alvo -code |
| Exemplo: ./reconsec full -url “https://testphp.vulnweb.com/” -code . –report final.html -out final.json |

## 4.6 Comando: version

Objetivo: Exibir a versão atual do binário e data de build. ./reconsec version

|  |
| --- |
| 5. Relatórios e Interpretação |

O ReconSec pode gerar três formatos de relatório:

1. JSON — formato padrão e mais detalhado.
2. TXT — formato legível, com resumo de resultados e data.
3. HTML — formato visual, ideal para apresentações e documentação.

Os relatórios incluem: - Cabeçalhos HTTP - Endpoints encontrados - Formulários e parâmetros - Versões de frameworks - Dependências identificadas - Pontos de atenção de segurança

|  |
| --- |
| 6. Boas Práticas e Ética Profissional |

O ReconSec deve ser usado apenas em ambientes e sistemas com autorização expressa do responsável.

Recomendações: - Sempre registrar escopo e permissão por escrito. - Evitar modos ativos em ambientes de produção. - Respeitar políticas de uso de rede e dados. - Guardar relatórios de forma segura (criptografia local). - Atualizar a ferramenta regularmente.

|  |
| --- |
| 7. Guia de Manutenção e Atualização |

Atualizar dependências: go mod tidy

Recompilar após atualizações: go build -o reconsec ./cmd/reconsec

Adicionar novos módulos: - Cada funcionalidade deve ser adicionada como novo pacote em pkg/. - Atualize internal/runner para incluir a chamada. - Atualize o help da CLI em internal/cli.

Melhorias recomendadas: - Implementar ML para priorização de vulnerabilidades. - Adicionar suporte a SARIF (para integração GitHub). - Implementar crawling headless opcional (chromedp). - Revisar templates HTML em web/report.tmpl.

Boas práticas de versionamento: - Criar branch “feature/” para novas funcionalidades. - Manter o changelog atualizado. - Testar antes de fazer push no repositório principal.

|  |
| --- |
| Apêndice — Resumo Rápido |

Comandos principais: - recon → reconhecimento passivo - sca → análise de dependências - scan → análise estática - test → testes dinâmicos seguros - full → pipeline completo - version → exibe a versão

Principais diretórios a editar: - pkg/recon/ → módulo de coleta de dados - pkg/sca/ → análise de dependências - pkg/report/ → relatórios - pkg/scanner/ → testes dinâmicos - internal/runner/ → fluxo geral de execução

|  |
| --- |
| Conclusão |

O ReconSec é uma ferramenta sólida para profissionais de AppSec que buscam precisão, controle e ética em suas análises. Sua arquitetura modular garante que cada componente possa ser aprimorado de forma independente, promovendo uma evolução contínua e segura.

Sempre que realizar alterações, registre, documente e teste. Segurança é um processo — não um destino.