Coyote Architecture Overview

A technical, shareable overview of the Coyote repository and agent security scanner, based on a codebase review. This document is designed to gather feedback on structure, flows, and extensibility.

Executive Summary

Coyote is a dual-purpose security tool that scans repositories for secrets and security smells, and analyzes AI agent configurations for risky capabilities. The repo contains a primarycoyotepackage with a Rich-based TUI and CLI, a scanning engine with pattern and entropy detection, reporting in multiple formats, baseline diffing, git history scanning, and webhook notifications. Agent analysis lives undercoyote/agents/and parallels a legacy or standalonemoltsec/package with similar functionality.

Goals

- Detect secrets, credentials, and security anti-patterns in code and history.
- Provide actionable reports in multiple formats for CI and humans.
- Track findings over time via stable IDs and baselines.
- Analyze AI agents for risky permissions and generate runtime policies.
- Offer a usable CLI and TUI for local workflows and watchers.

Non-Goals

- Full static analysis of complex code semantics.
- Live runtime sandboxing of arbitrary repositories.
- Remediation or automated secret rotation.

System Context

Coyote is run locally or in CI/CD. It consumes a repo path or a GitHub URL (viacoyote.shwatcher). Outputs are reports and optional webhook notifications. Agent analysis consumes agent config files and produces manifests, diffs, and policies.

Architecture Diagram

```
flowchart LR
user["User or CI"] --> cli["CLI/TUI: coyote/__main__.py + coyote/tui.py"]
```

```
cli --> scan["Scanner Engine"]
scan --> patterns["Pattern Rules"]
scan --> entropy["Entropy Detection"]
scan --> suppress["Suppression Rules"]
scan --> reports["Report Writers"]
scan --> baseline["Baseline Store"]
cli --> history["Git History Scan"]
cli --> notify["Webhook Notifications"]

user --> agentcli["Agent CLI: coyote/agents/__main__.py"]
agentcli --> agentanalyze["Agent Analyzer"]
agentcli --> agenttrack["Permission Tracker"]
agentcli --> agentpolicy["Policy Generator"]

reports --> outputs["JSON / Markdown / SARIF / HTML"]
baseline --> diff["Diff Summary"]
```

Core Flows

Repository Scan

- coyote/tui.pyparses arguments and loads config.

- coyote/scanner.pycollects files, filters exclusions, and checks size.

- coyote/patterns.pyapplies secret and smell regex rules.

- coyote/entropy.pyoptionally scans for high-entropy candidate strings.

- coyote/suppress.pyfilters findings using.coyote-ignore.

- Results are displayed in TUI or saved viacoyote/reporter.py.

Baseline Diff

- A scan is saved bycoyote/baseline.pyas.coyote-baseline.json.

- A new scan compares finding IDs to categorize new, fixed, and existing.

- coyote/tui.pyrenders a diff panel and can fail CI via--fail-on-new.

Git History Scan

- coyote/history.pyrunsgit log -pand parses added lines.

- Secret patterns are matched only against additions for each commit.

- Findings are grouped by commit with metadata for reporting.

Agent Analysis

- coyote/agents/__main__.pyloads an agent config file.

- Analyzer creates a capability manifest and risk summary.

- Tracker stores versions and computes diffs.

- Policy generator emits runtime policy JSON with strictness modes.

Key Modules and Responsibilities

Module	Responsibility
coyote/__main__.py	CLI entry and command routing
coyote/tui.py	Rich-based UI, CLI args, scan orchestration
coyote/scanner.py	File collection and pattern application
coyote/patterns.py	Secret and smell rules, sensitive filenames
coyote/entropy.py	Entropy-based secret detection
coyote/suppress.py	.coyote-ignoreparsing and filtering
coyote/reporter.py	JSON/Markdown/SARIF/HTML reports
coyote/baseline.py	Baseline save/load and diffing
coyote/history.py	Git history scanning
coyote/notifications.py	Slack/Discord notifications
coyote/agents/*	Agent analysis, diffing, policy
moltsec/*	Standalone agent analysis package

Data Model Overview

Finding

- finding_idis a stable hash of rule, file, line, and match value.

- Enables diffing across scans and suppression by ID.

Baseline

- JSON file storing findings and metadata.

- Diff uses set comparisons onfinding_id.

Agent Manifest

- Structured model with capabilities, risks, metadata, and summary.
- Used for diffing and policy generation.

Configuration

- config.yamloverrides defaults incoyote/config.py.
- config.example.yamlprovides a template for repo/branch, exclusions, outputs, and webhooks.

Extensibility

- Add rules tocoyote/patterns.pyfor new secret/smell types.
- Add new output formats incoyote/reporter.py.
- Extend agent capability detection incoyote/agents/analyzer.py.
- Add new notification channels incoyote/notifications.py.

Strengths

- Clear separation of detection, suppression, diffing, and reporting.
- Stable finding IDs enable reliable baselines and suppression.
- Multi-format outputs make CI and human review easy.
- Agent security analysis is integrated yet separable.

Risks and Opportunities

- Pattern-based detection may yield false positives and misses.
- History scanning parses diffs only and may miss context.
- Two agent analysis packages could create drift or duplication.
- Regex rules are centralized and can become a bottleneck for performance.

Open Questions

- Shouldmoltsec/be deprecated or promoted as a standalone package?

- Shouldcoyote/agents/andmoltsec/share a common core?

- Do we want configurable rule packs by ecosystem or language?

- Should entropy detection integrate context heuristics to reduce noise?

Suggested Next Steps

- Decide on the long-term relationship betweencoyote/agents/andmoltsec/.

- Define rule pack strategy and extension mechanisms.

- Consider performance profiling for large repos and history scans.

- Add tests around suppression and diff stability to guard regressions.

If you want a version tailored for a pitch deck or a shorter one-pager, I can create that too.