Cloak : Adversarial OSINT Evasion

Controlled, ethical experiments that inject harmless, deceptive artifacts into public signals (GitHub commits/repos + DNS records) to measure how OSINT pipelines index and propagate false indicators.

What It Does ?

Cloak makes safe, fake signals, checks OSINT sources to see if they notice them, records when and where they appear, measures how quickly and widely they spread, and creates easy-to-read charts and datasets for analysis.

Working

Setup & Config

- Config file and env variables store API keys, lab domain, and DB path.

Database

- Single SQLite DB using GORM.
- Tables:
    - `artifacts` : stores GitHub commits, repos, and DNS records.
    - `detections` : logs when artifacts are found in OSINT feeds.
    - `jobs` : manages tasks (generation, ingestion).
    - `audit_logs` : tracks every action with user, time, and reason.
- All writes are transactional; raw provider responses are stored for analysis.

Artifact Generation (GitHub)

- Use GitHub API to create repos under lab account.
- Prepare commits with fake but harmless metadata (author, timestamps).
- Store commit info in `artifacts` and log actions in `audit_logs` .
- Dry-run mode allows testing without creating anything.

Artifact Generation (DNS)

- Use Cloudflare API to create DNS records under lab domain (A, CNAME, TXT).
- Check records locally using `miekg/dns` .
- Store record info in `artifacts` and log actions.
- Teardown info saved for later deletion.

OSINT Ingestion

- Workers read artifacts and query OSINT providers: GitHub Search, VirusTotal, GreyNoise, OTX.
- Run queries in parallel using `errgroup` , respecting rate limits.
- Parse responses, normalize fields, and store in `detections` .
- Every API call is logged in `audit_logs` .

Normalization & Storage

- Map provider-specific data to a common format while keeping raw JSON.
- Allows checking metadata fidelity and cross-provider comparisons.

Web UI & Charts

- Gin serves API endpoints for artifacts, detections, and metrics.
- Chart.js displays timelines, detection rates, latency, and persistence.
- UI shows provider status, last query time, and audit logs.

Tech stack

- **Lang:** Go
- **HTTP:** `net/http` (+ Resty)
- **GitHub API**
- **DNS SDK:** `github.com/cloudflare/cloudflare-go`
- **DNS queries:** `github.com/miekg/dns`
- **DB:** SQLite
- **ORM**
- **Web server:** Gin
- **Charts:** Chart.js

Stack Usage

- **Go:** Backend (generators, workers, controllers, DB, web).
- **net/http /Resty:** call OSINT APIs, Cloudflare API, GitHub API; server web UI.
- **go-github:** create/manage lab repos, search commits/repos to measure index visibility.
- **cloudflare-go:** programmatically create/delete DNS A/TXT/CNAME records under lab domain.
- **miekg/dns:** local DNS validation & lookups.
- **SQLite:** store artifacts, detections, jobs, audit_logs.
- **Gin/Chi + Chart.js:** serve API + UI, render detection timelines and charts.
- **errgroup:** parallel queries with rate limiting and backoff.

Timeline

Week	Focus	Core Tasks	Deliverables
1	Prep & Infra	Team intro, project planning, stack discussion, define scope, setup GitHub & Cloudflare API keys, DB schema	Project charter, team roles, config samples, initial DB schema
2	Artifact Gen	GitHub commit spoofing & repo creation, Cloudflare DNS injection & teardown, unit tests	`github-generator`, `dns-injector`, sample artifacts in SQLite DB, artifact creation docs
3	OSINT Ingestion	Build adapters for GitHub Search, VirusTotal, GreyNoise; parallel queries, log ingestion	OSINT adapters, `detections` table, logging & audit, prototype web UI
4	Analysis	Compute metrics (visibility, latency, persistence), generate charts, refine experiments	Analysis scripts, charts & visualizations, updated UI reflecting results
5	Validation	Re-run experiments, robustness testing, finalize dataset, execute teardown, confirm ethics	Final redacted dataset, final report, audit log confirmation

Resources

- https://go.dev/ref/spec
- https://pkg.go.dev/golang.org/x/sync/errgroup
- https://pkg.go.dev/net
- https://github.com/go-resty/resty
- https://github.com/google/go-github
- https://pkg.go.dev/github.com/google/go-github/v50/github
- https://docs.github.com/en/rest/using-the-rest-api/getting-started-with-the-rest-api?apiVersion=2022-11-28
- https://github.com/cloudflare/cloudflare-go
- https://developers.cloudflare.com/api/resources/dns/
- https://github.com/miekg/dns
- https://github.com/mattn/go-sqlite3
- https://gorm.io/docs/
- https://github.com/gin-gonic/gin
- https://pkg.go.dev/golang.org/x/sync/errgroup
- https://www.chartjs.org/