Gitleaks Secret scanner

Source git history, directory, stdin → match → **Rules** default or .gitleaks.toml → filter → **Allowlists** paths, regexes, baseline, allow comments → report → **Findings** log, json, sarif, csv, junit

CONCEPTS

Exit code 1 means leaks were found, 0 means clean. That is what makes it usable as a CI or hook gate.

detect and protect are deprecated since v8.19.0 and hidden from --help. Use git, dir and stdin instead.

SCAN MODES

`gitleaks git`
Scan commit history of the current repo

`gitleaks git <repo-path>`
Scan another repository

`gitleaks dir <path>`
Scan a folder or single file

`cat <file> | gitleaks stdin`
Scan piped data

`gitleaks git --log-opts="<opts>"`
Pass git log options to limit commits

`gitleaks git --log-opts="--all"`
Include all branches and tags

SCAN MODES (cont.)

`-v / --verbose`
Print each finding in detail

`gitleaks version`
Show the installed version

REPORTS

`-r <file>`
Write findings to a report file

`-f json / csv / junit / sarif`
Choose the report format

`--redact`
Mask secrets in logs and stdout

`--exit-code <n>`
Exit code to use when leaks are found

`--no-banner`
Hide the ASCII banner

`--no-color`
Plain output for logs

BASELINE AND IGNORES

`gitleaks git -r baseline.json`
Create a baseline from today's findings

`-b <baseline.json>`
Report only findings not in the baseline

BASELINE AND IGNORES (cont.)

`// gitleaks:allow`
Comment on a line to skip its findings

`.gitleaksignore`
Repo file listing finding fingerprints to skip

CONFIGURATION

`-c <file>`
Use this config file

`.gitleaks.toml`
Config picked up from the scan target

`GITLEAKS_CONFIG=<file>`
Config path via environment

`--enable-rule <id>`
Run only the given rule IDs

`[extend] useDefault = true`
Keep default rules and add your own

`[extend] disabledRules = ["<id>"]`
Turn off inherited rules

`[extend] path = "<file>"`
Extend a custom config file

`[[rules]] id, regex, keywords`
Define a custom detection rule

ALLOWLISTS

`[[allowlists]]`
Global allowlist block

`paths = ['''<regex>''']`
Skip files by path regex

`regexes = ['''<regex>''']`
Skip findings matching a regex

`commits = ["<sha>"]`
Skip findings in given commits

`[[rules.allowlists]]`
Allowlist for a single rule

SCANNING OPTIONS

`-l <level>`
Log level: trace to fatal

HOOKS AND CI

`repo: https://github.com/gitleaks/gitleaks`
pre-commit repo entry; pin rev, id: gitleaks

`pre-commit install`
Activate the hook in this clone

`pre-commit autoupdate`
Bump the pinned gitleaks rev

HOOKS AND CI (cont.)

`SKIP=gitleaks git commit`
Skip the hook once

`gitleaks/gitleaks-action`
GitHub Action that runs the scan

`docker run -v <dir>:/path ghcr.io/gitleaks/gitleaks:latest git /path`
Run in Docker

`brew install gitleaks`
Install on macOS or Linuxbrew

RECIPES

Scan history and keep a redacted report

```
gitleaks git --redact -v \
  -f json -r gitleaks-report.json
```

Adopt on an old repo: baseline first, gate new leaks

```
gitleaks git -r baseline.json || true
gitleaks git -b baseline.json -r new.json
```

Allowlist test fixtures and docs (.gitleaks.toml)

```
[[allowlists]]
description = "fixtures"
paths = ['''testdata/.*''', '''docs/.*\.md''']
```