# Gitleaks Secret scanner

**START HERE**

**`gitleaks git -v`**
Scan the repo history

**`gitleaks dir -v <path>`**
Scan files, no git needed

**`cat <file> | gitleaks stdin -v`**
Scan piped data

**`-r report.json`**
Write a report file

**`-f sarif`**
Pick the report format

**`--redact`**
Hide secrets in output

**`-b report.json`**
Ignore known findings

**`gitleaks:allow`**
Comment to skip a line

**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.

Without a target, git and dir scan the current directory. A finding in old history stays a finding after you delete the file: rotate the secret.

## 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
*Anything git log accepts works, for example --all or a range such as commitA..commitB.*

## SCAN MODES (cont.)

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

**`-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

**`-f template --report-template <file>`**
Render with a Go template

**`--redact`**
Mask secrets in logs and stdout

**`--redact=<percent>`**
Mask only part of each secret

**`--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

**`// gitleaks:allow`**
Comment on a line to skip its findings
*Add it as a comment in whatever the file's syntax is, on the same line as the secret.*

**`--ignore-gitleaks-allow`**
Ignore gitleaks:allow comments

**`.gitleaksignore`**
Repo file listing finding fingerprints to skip
*Experimental. Put one fingerprint per line, copied from a finding's Fingerprint field.*

**`-i <path>`**
Path to .gitleaksignore or its folder

## CONFIGURATION

**`-c <file>`**
Use this config file

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

## CONFIGURATION (cont.)

**`GITLEAKS_CONFIG=<file>`**
Config path via environment

**`GITLEAKS_CONFIG_TOML=<toml>`**
Config content 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
*Cannot be combined with useDefault. Chaining works to a depth of 2.*

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

**`secretGroup / entropy`**
Pick the capture group; require randomness

**`path = '''<regex>'''`**
Restrict a rule to matching file paths

## ALLOWLISTS

**`[[allowlists]]`**
Global allowlist block
*Replaced the single [allowlist] table in v8.25.0.*

## ALLOWLISTS (cont.)

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

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

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

**`stopwords = ["<word>"]`**
Skip findings containing a word

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

## SCANNING OPTIONS

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

**`--max-target-megabytes <n>`**
Skip files bigger than this

**`--max-decode-depth <n>`**
Also scan decoded base64, hex, percent text

## 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

**`SKIP=gitleaks git commit`**
Skip the hook once

## HOOKS AND CI (cont.)

**`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
```

**Custom rule on top of the defaults (.gitleaks.toml)**

```
[extend]
useDefault = true

[[rules]]
id = "acme-token"
regex = '''acme_[a-z0-9]{32}'''
keywords = ["acme_"]
```

**Allowlist test fixtures and docs (.gitleaks.toml)**

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