Bandit Python security linter

CONCEPTS

Every finding has a severity (impact) and a confidence (how sure Bandit is). Triage high/high first.

Python files parsed to AST
↓ plugins
Tests B1xx-B7xx selected by -t / -s
↓ filters
Issues severity + confidence
↓ -f
Report screen, json, html...

Bandit exits 1 when it reports issues, so it works as a CI gate; --exit-zero overrides that.

Test IDs group by theme: B1xx misc, B3xx calls, B4xx imports, B5xx crypto/TLS, B6xx injection, B7xx templates.

SCANNING

`bandit -r <dir>`
Scan a directory recursively

`bandit <file.py> ...`
Scan specific files

`bandit --version`
Show Bandit and Python version

`bandit -r <dir> -x <paths>`
Exclude comma-separated paths

`bandit -r <dir> -v`
Verbose: list files scanned and excluded

`bandit -r <dir> -q`
Quiet: only print on errors

`bandit -r <dir> -n <n>`
Max code lines shown per issue

`bandit -r <dir> --exit-zero`
Always exit 0

FILTERING

`bandit -r <dir> -l / -ll / -lll`
Severity low / medium / high and up

`bandit -r <dir> -i / -ii / -iii`
Confidence low / medium / high and up

FILTERING (cont.)

`bandit -r <dir> --severity-level <level>`
Severity as a word: all, low, medium, high

`bandit -r <dir> --confidence-level <level>`
Confidence as a word

`bandit -r <dir> -t <ids>`
Run only these test IDs

`bandit -r <dir> -s <ids>`
Skip these test IDs

`bandit -r <dir> -p <profile>`
Use a named profile from the config

OUTPUT

`bandit -r <dir> -f json`
Machine-readable report

`bandit -r <dir> -f <format>`
csv, html, json, screen, txt, xml, yaml, custom

`bandit -r <dir> -o <file>`
Write the report to a file

`bandit -r <dir> -f custom --msg-template "{relpath}: {line}: {test_id} {msg}"`
One line per issue

`bandit -r <dir> -f html -o <file>`
Browsable HTML report

CONFIGURATION

`bandit -r <dir> -c <file>`
Use a config file

`bandit-config-generator -o <file>`
Generate a YAML config to edit

pyproject.toml

```
[tool.bandit]
exclude_dirs = ["tests", ".venv"]
skips = ["B101"]
```

SUPPRESSING

`# nosec`
Ignore all findings on this line

`# nosec B602`
Ignore only the named test(s)

`bandit -r <dir> --ignore-nosec`
Report lines marked nosec anyway

SUPPRESSING (cont.)

`bandit -r <dir> -b <baseline.json>`
Show only issues not in the baseline

Put the nosec comment on the line the issue is reported at; for multi-line calls that is the first line.

COMMON FINDINGS

Tests you will meet most

ID	Flags
B101	assert (stripped under -O)
B105-B107	hardcoded password
B108	hardcoded /tmp path
B113	request without timeout
B301	pickle load
B311	random for security use
B324	weak hash (MD5, SHA1)
B506	yaml.load unsafe
B602	subprocess with shell=True
B608	SQL built from strings

Fix directions

Finding	Use instead
shell=True	argument list, no shell
yaml.load	yaml.safe_load
random	secrets module
MD5/SHA1	SHA-256 or usedforsecurity=False

RECIPES

CI gate on meaningful issues

```
bandit -r src -ll -ii -q
# exit 1 fails the job
```

Adopt on a legacy codebase

```
bandit -r src -f json -o
baseline.json --exit-zero
bandit -r src -b baseline.json
```

Pre-commit hook

```
# .pre-commit-config.yaml
- repo: https://github.com/PyCQA/
bandit
  rev: <tag>
  hooks:
    - id: bandit
      args: ["-c",
"pyproject.toml"]
      additional_dependencies:
["bandit[toml]"]
```

Targeted review

```
bandit -r src -t B602,B605,B608 -
f custom \
  --msg-template "{relpath}:
{line} {test_id}"
```