

django

github.com/django/django

Scanned 29 October 2026 at 09:23 · Report generated 29 October 2026 · Python · 91,537 stars

OVERALL ASSESSMENT

Critical

Combined score 493 · 296 findings after filtering

Summary

147 findings likely need attention, 90 look like noise. Most notable: Detects dynamic code loading: downloading and importing/executing code at runtime [thre... (99% confidence).

Generated from finding counts. A plain-English summary was not available for this scan.

Risk, by type

MalDet scores two different questions separately, because they call for different responses: code can be badly written without being hostile, and hostile code is not always badly written.

VULNERABILITY

Critical (437)

Insecure coding an attacker could exploit. Usually mistakes, and usually fixable in place.

MALICIOUS PATTERN

High (56)

Signs the code may be trying to do harm: hidden payloads, credential access, persistence, exfiltration.

What each detector found

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
bandit	Insecure coding patterns in Python source	0	170	96	266
yara	Signatures of malicious behaviour in source files	8	0	5	13
dep_checker	Vulnerable, unpinned or typosquatted dependencies	0	1	8	9
guarddog	Malicious package and install-time behaviour	5	0	3	8

147 of 296 findings were rated more likely genuine than noise by the classifier.

Evidence

Identical findings are grouped, so a single rule firing many times takes one row. Ordered worst first. “Seen” is how many times that finding occurred.

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	guarddog	Detects dynamic code loading: downloading and importing/executing code at runtime [threat-runtime-dynamic-loader]	1	/django/contrib/auth/hashers.py:127
HIGH	guarddog	Detects download-and-execute patterns: fetching a remote file then executing it [threat-process-download-exec]	3	/scripts/manage_translations.py:404
HIGH	guarddog	Detects steganography decode followed by code execution [threat-runtime-obfuscation-steganography]	1	/django/template/defaulttags.py:919
HIGH	yara	YARA rule matched: detect_credential_harvester — References credential/secret material by name AND a deliberate outbound-send API in the same file - narrower than 'reads any env var', which large legitimate codebases do constantly regardless of intent. Still not proof the two are connected (YARA c	1	/scripts/pr_quality/check_pr.py
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	2	/scripts/do_django_release.py
HIGH	yara	YARA rule matched: detect_obfuscated_string_execution — Reconstructs a string character-by-character or from hex escapes, then immediately executes it - fromCharCode/chr() alone are common in ordinary text-processing code, so this requires both the reconstruction and an execution call together	4	/django/contrib/admin/static/admin/js/vendor/jquery/jquery.min.js
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	1	/extras/django_bash_completion

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Using xml.dom.pulldom.parse to parse untrusted XML data is known to be vulnerable to XML attacks. Replace xml.dom.pulldom.parse with its defusedxml equivalent function or make sure defusedxml.defuse_stdlib() is called	1	/django/core/serializers/xml_serializer.py:244
MEDIUM	bandit	Potential XSS on mark_safe function.	56	/django/template/defaultfilters.py:621
MEDIUM	bandit	Use of mark_safe() may expose cross-site scripting vulnerabilities and should be reviewed.	51	/django/utils/translation/trans_real.py:393
MEDIUM	bandit	Use of RawSQL potential SQL attack vector.	6	/django/db/models/sql/query.py:2622
MEDIUM	bandit	Use of extra potential SQL attack vector.	1	/django/db/models/fields/related_descriptors.py:1241
MEDIUM	dep_checker	Unpinned dependency: 'tzdata' has no version lock — supply chain risk	1	pyproject.toml
MEDIUM	bandit	Use of weak SHA1 hash for security. Consider usedforsecurity=False	2	/django/template/loaders/cached.py:96
MEDIUM	bandit	Use of weak MD5 hash for security. Consider usedforsecurity=False	2	/django/contrib/auth/hashers.py:665
MEDIUM	bandit	Audit url open for permitted schemes. Allowing use of file:/ or custom schemes is often unexpected.	2	/scripts/pr_quality/check_pr.py:122
MEDIUM	bandit	Possible SQL injection vector through string-based query construction.	35	/django/core/cache/backends/db.py:302

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Function call with shell=True parameter identified, possible security issue.	1	/scripts/ manage_translations.py: 202
MEDIUM	bandit	Use of possibly insecure function - consider using safer ast.literal_eval.	1	/scripts/ manage_translations.py: 404
MEDIUM	bandit	Use of exec detected.	3	/django/ core/ management/ commands/ shell.py:89
MEDIUM	bandit	Pickle and modules that wrap it can be unsafe when used to deserialize untrusted data, possible security issue.	7	/django/ core/cache/ backends/ locmem.py:43
MEDIUM	bandit	Possible binding to all interfaces.	1	/django/ core/ management/ commands/ runserver.py: 178
MEDIUM	bandit	Call to requests without timeout	1	/scripts/ manage_translations.py: 76
LOW	bandit	Using pulldom to parse untrusted XML data is known to be vulnerable to XML attacks. Replace pulldom with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	1	/django/ core/ serializers/ xml_serializer.py:6
LOW	bandit	Using handler to parse untrusted XML data is known to be vulnerable to XML attacks. Replace handler with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	1	/django/ core/ serializers/ xml_serializer.py:7
LOW	bandit	Using ExpatParser to parse untrusted XML data is known to be vulnerable to XML attacks. Replace ExpatParser with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	1	/django/ core/ serializers/ xml_serializer.py:8

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Using XMLGenerator to parse untrusted XML data is known to be vulnerable to XML attacks. Replace XMLGenerator with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	1	/django/ utils/ xmlutils.py: 5
LOW	bandit	Consider possible security implications associated with PicklingError module.	1	/django/ urls/ resolvers.py :14
LOW	bandit	Possible hardcoded password: 'POST'	2	/django/ middleware/ csrf.py:345
LOW	bandit	Possible hardcoded password: 'COLLATE'	1	/django/db/ backends/ sqlite3/ introspecti on.py:450
LOW	bandit	Possible hardcoded password: 'not'	1	/django/ template/ smartif.py:1 74
LOW	bandit	Possible hardcoded password: '_password_reset_token'	1	/django/ contrib/ auth/ views.py:246
LOW	bandit	Possible hardcoded password: 'is not'	1	/django/ template/ smartif.py:1 72
LOW	bandit	Possible hardcoded password: 'not in'	1	/django/ template/ smartif.py:1 75
LOW	bandit	Possible hardcoded password: 'is'	1	/django/ template/ smartif.py:1 71
LOW	bandit	Possible hardcoded password: 'NOTPROVIDED'	1	/django/ template/ defaulttags. py:79

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Possible hardcoded password: 'usable_password'	1	/django/contrib/auth/forms.py:166
LOW	bandit	Possible hardcoded password: 'password1'	1	/django/contrib/auth/forms.py:131
LOW	bandit	Possible hardcoded password: 'password2'	1	/django/contrib/auth/forms.py:123
LOW	bandit	Possible hardcoded password: '!'	1	/django/contrib/auth/hashers.py:25
LOW	bandit	Possible hardcoded password: 'ALTER USER %(user)s IDENTIFIED BY "%(password)s"'	1	/django/db/backends/oracle/creation.py:292
LOW	bandit	Possible hardcoded password: 'django-insecure-'	1	/django/core/checks/security/base.py:22
LOW	bandit	Possible hardcoded password: 'set-password'	1	/django/contrib/auth/views.py:262
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	3	/scripts/do_django_release.py:142
LOW	bandit	Possible hardcoded password: 'CSRF token missing.'	1	/django/middleware/csrf.py:33
LOW	bandit	Possible hardcoded password: 'test-token'	1	/scripts/pr_limit_tests.py:10
LOW	bandit	Possible hardcoded password: ''	4	/django/conf/global_settings.py:215

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	subprocess call with shell=True seems safe, but may be changed in the future, consider rewriting without shell	1	/django/ utils/ version.py:237
LOW	bandit	Consider possible security implications associated with the subprocess module.	11	/django/ core/ management/ commands/ dbshell.py:1
LOW	bandit	Starting a process with a partial executable path	5	/scripts/ do_django_release.py:157
LOW	bandit	subprocess call - check for execution of untrusted input.	15	/scripts/ do_django_release.py:202
LOW	dep_checker	Loose version constraint: 'qunit^2.25.0' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'sqlparse>=0.5.0' — attacker could push malicious update	1	pyproject.toml
LOW	dep_checker	Loose version constraint: 'grunt^1.6.2' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'grunt-contrib-qunit^10.2.0' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'grunt-cli^1.5.0' — attacker could push malicious update	1	package.json
LOW	dep_checker	Loose version constraint: 'bcrypt>=4.1.1' — attacker could push malicious update	1	pyproject.toml
LOW	dep_checker	Loose version constraint: 'asgiref>=3.13' — attacker could push malicious update	1	pyproject.toml
LOW	dep_checker	Loose version constraint: 'argon2-cffi>=23.1.0' — attacker could push malicious update	1	pyproject.toml
LOW	yara	YARA rule matched: detect_downloader — Downloads a remote resource - common and low-signal alone; see detect_download_and_execute for the composite version	5	/django/ core/ management/ templates.py

Showing the 60 most significant of 61 distinct findings. The counts in the previous section cover all of them.

How to read this report

- **This is a static scan.** The code was read, never executed. Behaviour that only appears at runtime is outside what any of these detectors can see.
- **Findings are not all equally real.** A classifier rates each finding on how likely it is to be genuine rather than noise, and that rating weights the scores. Measured against a hand-reviewed sample it reached precision 0.52 and recall 0.62 — useful for ranking, not a verdict on any single finding.
- **A high score is not proof of malice.** Security tools, penetration-testing frameworks and malware research repositories contain the same patterns as real malware by design. The two risk types above exist to keep that distinction visible.
- **Absence of findings is not proof of safety.** A detector that is not installed, or a language none of them parse, produces silence rather than a warning.
- **This scan is a snapshot.** The repository may have changed since 29 October 2026.