

core

github.com/opnsense/core

Scanned 24 September 2026 at 07:24 · Report generated 29 October 2026 · PHP · 4,696 stars

OVERALL ASSESSMENT

Critical

Combined score 564 · 493 findings after filtering

Summary

This repository has two distinct security issues that require immediate attention before it can be used in production. The most serious problem is the high vulnerability risk score of 560, which indicates that the code contains dangerous patterns like untrusted input being passed directly to shell commands and SQL queries without proper protection. These findings are not signs of malicious intent but rather examples of common insecure coding practices that an attacker could exploit to compromise the system. While there are some lower-priority alerts about potential obfuscation or credential reading, they do not match the severity of the critical vulnerabilities found elsewhere in the codebase.

Written by a language model running locally on the scanning machine, from the findings in this report. The risk levels above and below were decided by the scoring model, not by this summary.

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 (560)

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

MALICIOUS PATTERN

Low (4)

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	7	51	252	310
semgrep	Insecure coding patterns across many languages	69	88	1	158

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	16	0	6	22
guarddog	Malicious package and install-time behaviour	0	0	3	3

68 of 493 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	semgrep	linux shadow file detected	1	/src/etc/rc.subr.d/recover:82
HIGH	semgrep	Untrusted input might be used to build an HTTP request, which can lead to a Server-side request forgery (SSRF) vulnerability. SSRF allows an attacker to send crafted requests from the server side to other internal or external systems. SSRF can lead to unauthorized access to sensitive data and, in so	1	/src/www/diag_backup.php:316
HIGH	semgrep	Found 'subprocess' function 'run' with 'shell=True'. This is dangerous because this call will spawn the command using a shell process. Doing so propagates current shell settings and variables, which makes it much easier for a malicious actor to execute commands. Use 'shell=False' instead.	3	/src/opnsense/scripts/filter/kill_states.py:51
HIGH	semgrep	Found 'subprocess' function 'Popen' with 'shell=True'. This is dangerous because this call will spawn the command using a shell process. Doing so propagates current shell settings and variables, which makes it much easier for a malicious actor to execute commands. Use 'shell=False' instead.	1	/src/opnsense/service/modules/actions/stream_output.py:53

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	Avoiding SQL string concatenation: untrusted input concatenated with raw SQL query can result in SQL Injection. In order to execute raw query safely, prepared statement should be used. SQLAlchemy provides TextualSQL to easily used prepared statement with named parameters. For complex SQL composition	11	/src/ opnsense/ scripts/ unbound/ logger.py:103
HIGH	semgrep	bcrypt hash detected	1	/src/etc/ config.xml. sample:24
HIGH	semgrep	Executing non-constant commands. This can lead to command injection.	51	/src/etc/ rc.newwanip: 105
HIGH	bandit	By default, jinja2 sets autoescape to False. Consider using autoescape=True or use the select_autoescape function to mitigate XSS vulnerabilities.	1	/src/ opnsense/ service/ modules/ template.py: 58
HIGH	bandit	Starting a process with a shell, possible injection detected, security issue.	2	/src/ opnsense/ site-python/ sqlite3_hel per.py:79
HIGH	bandit	subprocess call with shell=True identified, security issue.	4	/src/ opnsense/ service/ modules/ actions/ script.py:40
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	1	/src/ opnsense/ scripts/ openvpn/ genkey.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	/src/ opnsense/ mvc/app/ controllers/ OPNsense/ Monit/Api/ StatusContr oller.php

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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	9	/src/ opnsense/ scripts/ captiveportal/ htdocs_default/js/ jquery-3.5.1.min.js
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	4	/src/etc/ inc/ system.inc
MEDIUM	semgrep	RegExp() called with a `search` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controll	1	/src/ opnsense/ www/js/ tokenize2.js:654
MEDIUM	semgrep	RegExp() called with a `map` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled	2	/src/ opnsense/ www/js/ bootstrap-select.js:568
MEDIUM	semgrep	RegExp() called with a `str` function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled	1	/src/ opnsense/ www/js/ polyfills.js:168
MEDIUM	semgrep	These permissions `0o666` are widely permissive and grant access to more people than may be necessary. A good default is `0o644` which gives read and write access to yourself and read access to everyone else.	1	/src/ opnsense/ service/ modules/ processhandler.py:97

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	These permissions <code>`root_perm & (~stat.S_IXGRP & ~stat.S_IXUSR & ~stat.S_IXOTH)`</code> are widely permissive and grant access to more people than may be necessary. A good default is <code>`0o644`</code> which gives read and write access to yourself and read access to everyone else.	1	/src/opnsense/service/modules/template.py:292
MEDIUM	semgrep	Found direct access to a PHP variable without HTML escaping inside an inline PHP statement setting data from <code>`\$_REQUEST[...]'`</code> . When untrusted input can be used to tamper with a web page rendering, it can lead to a Cross-site scripting (XSS) vulnerability. XSS vulnerabilities occur when untrusted input	20	/src/www/status_wireless.php:108
MEDIUM	bandit	Using <code>xml.etree.cElementTree.parse</code> to parse untrusted XML data is known to be vulnerable to XML attacks. Replace <code>xml.etree.cElementTree.parse</code> with its defusedxml equivalent function or make sure <code>defusedxml.defuse_stdlib()</code> is called	1	/src/opnsense/service/modules/config.py:54
MEDIUM	bandit	Chmod setting a permissive mask <code>0o666</code> on file (<code>socket_filename</code>).	1	/src/opnsense/service/modules/processhandler.py:97
MEDIUM	bandit	Using <code>xml.etree.ElementTree.parse</code> to parse untrusted XML data is known to be vulnerable to XML attacks. Replace <code>xml.etree.ElementTree.parse</code> with its defusedxml equivalent function or make sure <code>defusedxml.defuse_stdlib()</code> is called	1	/src/opnsense/scripts/captiveportal/lib/__init__.py:100
MEDIUM	semgrep	Make sure comparisons involving md5 values are strict (use <code>`===`</code> not <code>`==`</code>) to avoid type juggling issues	2	/src/opnsense/scripts/openvpn/ovpn_service_control.php:104

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Detected possible formatted SQL query. Use parameterized queries instead.	5	/src/ opnsense/ scripts/ suricata/ lib/ rulecache.py :446
MEDIUM	bandit	Using xml.etree.ElementTree.fromstring to parse untrusted XML data is known to be vulnerable to XML attacks. Replace xml.etree.ElementTree.fromstring with its defusedxml equivalent function or make sure defusedxml.defuse_stdlib() is called	4	/src/ opnsense/ scripts/ interfaces/ list_regdom ain.py:40
MEDIUM	semgrep	The special variable IFS affects how splitting takes place when expanding unquoted variables. Don't set it globally. Prefer a dedicated utility such as 'cut' or 'awk' if you need to split input data. If you must use 'read', set IFS locally using e.g. 'IFS="," read -a my_array'.	1	/src/ opnsense/ scripts/ firmware/ license.sh:3 5
MEDIUM	semgrep	Detected direct use of jinja2. If not done properly, this may bypass HTML escaping which opens up the application to cross-site scripting (XSS) vulnerabilities. Prefer using the Flask method 'render_template()' and templates with a '.html' extension in order to prevent XSS.	1	/src/ opnsense/ service/ modules/ template.py: 58
MEDIUM	bandit	Use of weak MD5 hash for security. Consider usedforsecurity=False	11	/src/ opnsense/ scripts/ suricata/ lib/ downloader. py:161
MEDIUM	semgrep	Untrusted user input in `importlib.import_module()` function allows an attacker to load arbitrary code. Avoid dynamic values in `importlib.import_module()` or use a whitelist to prevent running untrusted code.	3	/src/ opnsense/ scripts/ syslog/ logformats/ __init__.py: 153

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Possible SQL injection vector through string-based query construction.	11	/src/ opnsense/ scripts/ suricata/ lib/ rulecache.py :268
MEDIUM	bandit	Possible binding to all interfaces.	4	/src/ opnsense/ scripts/ unbound/ unbound_wat cher.py:74
MEDIUM	semgrep	Calling `unserialize()` with user input in the pattern can lead to arbitrary code execution. Consider using JSON or structured data approaches (e.g. Google Protocol Buffers).	3	/src/www/ interfaces. php:327
MEDIUM	bandit	Probable insecure usage of temp file/directory.	13	/src/ opnsense/ scripts/ openvpn/ ovpn_event. py:78
MEDIUM	bandit	Call to requests without timeout	5	/src/ opnsense/ scripts/ filter/lib/ alias/ uri.py:79
MEDIUM	semgrep	Using user input when deleting files with `unlink()` is potentially dangerous. A malicious actor could use this to modify or access files they have no right to.	47	/src/ opnsense/ scripts/ openvpn/ ovpn_servic e_control.p hp:94
LOW	bandit	Using xml.etree.cElementTree to parse untrusted XML data is known to be vulnerable to XML attacks. Replace xml.etree.cElementTree with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	3	/src/ opnsense/ service/ modules/ config.py:35

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	guarddog	Detects reading of environment variables (credential access, often contains secrets) [threat-runtime-environment-read]	3	/src/opnsense/scripts/openvpn/ovpn_event.py:46
LOW	bandit	Using ElementTree to parse untrusted XML data is known to be vulnerable to XML attacks. Replace ElementTree with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	1	/src/opnsense/scripts/health/fetchData.py:37
LOW	bandit	Using xml.etree.ElementTree to parse untrusted XML data is known to be vulnerable to XML attacks. Replace xml.etree.ElementTree with the equivalent defusedxml package, or make sure defusedxml.defuse_stdlib() is called.	4	/src/opnsense/scripts/captiveportal/lib/__init__.py:29
LOW	semgrep	Detected string concatenation with a non-literal variable in a util.format / console.log function. If an attacker injects a format specifier in the string, it will forge the log message. Try to use constant values for the format string.	1	/src/opnsense/www/js/opnsense_widget_manager.js:588
LOW	bandit	Try, Except, Continue detected.	1	/src/opnsense/scripts/filter/lib/states.py:170
LOW	bandit	subprocess call with shell=True seems safe, but may be changed in the future, consider rewriting without shell	1	/src/opnsense/scripts/system/swapinfo.py:32
LOW	bandit	Consider possible security implications associated with the subprocess module.	79	/src/opnsense/scripts/filter/lib/alias/interface.py:28

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
LOW	bandit	Starting a process with a partial executable path	12	/src/ opnsense/ scripts/ syslog/ log_matcher. py:64
LOW	bandit	subprocess call - check for execution of untrusted input.	151	/src/ opnsense/ scripts/ filter/lib/ alias/ auth.py:41
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	6	/src/ opnsense/ scripts/ suricata/ lib/ downloader. py

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 24 September 2026.