

ondemand

github.com/OSC/ondemand

Scanned 24 September 2026 at 06:54 · Report generated 29 October 2026 · JavaScript · 514 stars

OVERALL ASSESSMENT

Critical

Combined score 760 · 297 findings after filtering

Summary

This repository has two major security issues that require immediate attention before it can be used in production. The most critical finding is a high vulnerability risk score of 716.0, which indicates the code contains dangerous patterns like SQL injection and insecure file downloads that an attacker could exploit to steal data or run malicious scripts. While there are also signs of potential malicious intent flagged by tools like YARA and GuardDog, these findings appear to be related to known malware techniques rather than intentional attacks on your own system. However, the code still contains several other serious problems, including a high score of 44.0 for suspicious patterns that might indicate an attempt to bypass security checks or execute remote files. Because of these combined issues, you should not use this repository in any production environment without first fixing all the identified vulnerabilities and removing the dangerous code patterns.

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 (716)

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

MALICIOUS PATTERN

High (44)

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
semgrep	Insecure coding patterns across many languages	6	256	0	262
yara	Signatures of malicious behaviour in source files	8	1	19	28
guarddog	Malicious package and install-time behaviour	3	4	0	7

264 of 297 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	Untrusted input might be used to build a database query, which can lead to a SQL injection vulnerability. An attacker can execute malicious SQL statements and gain unauthorized access to sensitive data, modify, delete data, or execute arbitrary system commands. To prevent this vulnerability, use pre	1	/apps/ dashboard/ app/ controllers/ widgets_con troller.rb:1 3
HIGH	guarddog	Detects download-and-execute patterns: fetching a remote file then executing it [threat-process-download-exec]	3	/apps/ dashboard/ app/ javascript/ files/ file_ops.js: 402
HIGH	semgrep	Generic Secret detected	1	/ood-portal- generator/ share/ ood_portal_ example.yml: 425
HIGH	semgrep	Insecure WebSocket Detected. WebSocket Secure (wss) should be used for all WebSocket connections.	1	/ mod_ood_pro xy/lib/ood/ proxy.lua:13
HIGH	semgrep	bcrypt hash detected	3	/ood-portal- generator/ spec/ application_ spec.rb:312

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	1	/apps/shell/public/javascripts/hterm_all_1.92.1.mod_1.js
HIGH	yara	YARA rule matched: detect_download_and_execute — Downloads a remote resource AND executes it - the actually dangerous combination, not just downloading alone	1	/lib/tasks/test.rb
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	2	/apps/dashboard/public/noVNC-1.3.0/legacy/app.js
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	4	/spec/e2e/00_package_spec.rb
MEDIUM	semgrep	RegExp() called with a `prefix` 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	/apps/dashboard/app/javascript/dynamic_forms.js:130
MEDIUM	semgrep	RegExp() called with a `optionForType` 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-c	1	/apps/dashboard/app/javascript/dynamic_forms.js:938

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	'raw' renders raw HTML, as the name implies. This means that normal HTML escaping is bypassed. If user data can be controlled here, this exposes your application to cross-site scripting (XSS). If you need to do this, be sure to correctly sanitize the data using a library such as DOMPurify.	3	/apps/ dashboard/ app/views/ support_ticket/ email_service_template. html.erb:5
MEDIUM	semgrep	'html_safe' renders raw HTML. This means that normal HTML escaping is bypassed. If user data can be controlled here, this exposes your application to cross-site scripting (XSS). If you need to do this, be sure to correctly sanitize the data using a library such as DOMPurify.	2	/apps/ myjobs/app/ views/ layouts/ application. html.erb:38
MEDIUM	semgrep	Detected a template variable used in a script tag. Although template variables are HTML escaped, HTML escaping does not always prevent cross-site scripting (XSS) attacks when used directly in JavaScript. If you need to do this, use `escape_javascript` or its alias, `j`. However, this will not protect	8	/apps/ dashboard/ app/views/ layouts/ application. html.erb:13
MEDIUM	semgrep	Found an unscoped `find(...)` with user-controllable input. If the ActiveRecord model being searched against is sensitive, this may lead to Insecure Direct Object Reference (IDOR) behavior and allow users to read arbitrary records. Scope the find to the current user, e.g. `current_user.accounts.find`	17	/apps/ dashboard/ app/ controllers/ products_controller.rb: 65
MEDIUM	semgrep	Detected an explicit unescape in a Mustache template, using triple braces '{{{...}}}' or ampersand '&'. If external data can reach these locations, your application is exposed to a cross-site scripting (XSS) vulnerability. If you must do this, ensure no external data can reach this location.	1	/apps/shell/ views/ index.hbs:90

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	'content_tag' exhibits unintuitive escaping behavior and may accidentally expose your application to cross-site scripting. If using Rails 2, only attribute values are escaped. If using Rails 3, content and attribute values are escaped. Tag and attribute names are never escaped. Because of this, it i	21	/apps/ dashboard/ app/views/ dashboard/ _motd_rss.h tml.erb:6
MEDIUM	semgrep	'html_safe()' does not make the supplied string safe. 'html_safe()' bypasses HTML escaping. If external data can reach here, this exposes your application to cross-site scripting (XSS) attacks. Ensure no external data reaches here.	17	/apps/ dashboard/ app/helpers/ products_he lper.rb:59
MEDIUM	semgrep	Detected non-static command inside Open3.capture2e. Audit the input to 'Open3.capture2e'. If unverified user data can reach this call site, this is a code injection vulnerability. A malicious actor can inject a malicious script to execute arbitrary code.	2	/apps/ dashboard/ app/models/ product.rb:2 79
MEDIUM	semgrep	Detected non-static command inside Open3.capture3. Audit the input to 'Open3.capture3'. If unverified user data can reach this call site, this is a code injection vulnerability. A malicious actor can inject a malicious script to execute arbitrary code.	2	/apps/ myjobs/app/ models/ filesystem. rb:79
MEDIUM	semgrep	Detected non-static command inside Open3.popen2e. Audit the input to 'Open3.popen2e'. If unverified user data can reach this call site, this is a code injection vulnerability. A malicious actor can inject a malicious script to execute arbitrary code.	1	/apps/ dashboard/ app/ controllers/ products_co ntroller.rb: 122
MEDIUM	semgrep	Detected non-static command inside Open3.popen3. Audit the input to 'Open3.popen3'. If unverified user data can reach this call site, this is a code injection vulnerability. A malicious actor can inject a malicious script to execute arbitrary code.	1	/apps/ dashboard/ lib/ rclone_util. rb:319

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Checks for calls to <code>without_protection</code> during mass assignment (which allows record creation from hash values). This can lead to users bypassing permissions protections. For Rails 4 and higher, mass protection is on by default. Fix: Don't use <code>:without_protection => true</code> . Instead, configure <code>attr_access</code>	2	<code>/apps/myjobs/app/controllers/templates_controller.rb:14</code>
MEDIUM	semgrep	' <code>content_tag()</code> ' bypasses HTML escaping for some portion of the content. If external data can reach here, this exposes your application to cross-site scripting (XSS) attacks. Ensure no external data reaches here. If you must do this, create your HTML manually and use ' <code>html_safe</code> '. Ensure no external d	27	<code>/apps/dashboard/app/helpers/dashboard_helper.rb:97</code>
MEDIUM	semgrep	Should not use SHA1 to generate hashes. There is a proven SHA1 hash collision by Google, which could lead to vulnerabilities. Use SHA256, SHA3 or other hashing functions instead.	2	<code>/apps/dashboard/app/models/batch_connect/session.rb:648</code>
MEDIUM	semgrep	Detected a unquoted template variable as an attribute. If unquoted, a malicious actor could inject custom JavaScript handlers. To fix this, add quotes around the template expression, like this: " <code><%= expr %></code> ".	20	<code>/apps/dashboard/app/views/batch_connect/session_contexts/_form.html.erb:26</code>
MEDIUM	semgrep	Using user input when accessing files is potentially dangerous. A malicious actor could use this to modify or access files they have no right to.	4	<code>/apps/dashboard/app/controllers/products_controller.rb:119</code>
MEDIUM	semgrep	This Dependabot configuration does not set a cooldown period. Newly published packages can be malicious or unstable. Add a <code>`cooldown`</code> block with <code>`default-days: 7`</code> to each <code>`package-ecosystem`</code> entry under <code>`updates`</code> to wait 7 days before proposing updates to newly published package versions. Reference:	21	<code>/.github/dependabot.yml:12</code>

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Detected a template variable used in an anchor tag with the 'href' attribute. This allows a malicious actor to input the 'javascript:' URI and is subject to cross- site scripting (XSS) attacks. If using a relative URL, start with a literal forward slash and concatenate the URL, like this: href='/<%=	23	/apps/ myjobs/app/ views/ layouts/ application. html.erb:36
MEDIUM	semgrep	When a redirect uses user input, a malicious user can spoof a website under a trusted URL or access restricted parts of a site. When using user-supplied values, sanitize the value before using it for the redirect.	18	/apps/ dashboard/ app/ controllers/ products_co ntroller.rb: 96
MEDIUM	yara	YARA rule matched: detect_clipboard_hijack — Reads the clipboard alongside crypto-wallet-address-shaped patterns - a known 'clipper' malware technique that swaps a copied address for the attacker's own	1	/apps/shell/ public/ javascripts/ hterm_all_1. 92.1.mod_1. js
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	4	/apps/ dashboard/ app/ controllers/ products_co ntroller.rb: 84
MEDIUM	semgrep	Found that the setting for providing detailed exception reports in Rails is set to true. This can lead to information exposure, where sensitive system or internal information is displayed to the end user. Instead, turn this setting off.	4	/apps/ myjobs/ config/ environments / development. rb:15
MEDIUM	semgrep	RegExp() called with a `name` 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	/apps/ dashboard/ public/ noVNC-1.3.0/ app/ webutil.js:3 4

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Detected manual creation of an ERB template. Manual creation of templates may expose your application to server-side template injection (SSTI) or cross-site scripting (XSS) attacks if user input is used to create the template. Instead, create a '.erb' template file and use 'render'.	11	/apps/ dashboard/ app/models/ batch_connect/ session.rb:81
MEDIUM	semgrep	Detected non-static command inside `...`. If unverified user data can reach this call site, this is a code injection vulnerability. A malicious actor can inject a malicious script to execute arbitrary code.	12	/apps/ myjobs/app/ models/ filesystem. rb:88
MEDIUM	semgrep	GitHub Actions step uses a mutable tag or branch reference. Tags and branch names can be silently repointed by the action owner, enabling supply-chain attacks — as seen in the trivy-action and kics-github-action compromises. Pin the reference to a full 40-character commit SHA instead, e.g. `uses: ac	29	/.github/ workflows/ tests.yml:90
MEDIUM	semgrep	Checks for any usage of http servers instead of https servers. Encourages the usage of https protocol instead of http, which does not have TLS and is therefore unencrypted. Using http can lead to man-in-the-middle attacks in which the attacker is able to read sensitive information.	1	/apps/shell/ app.js:94
MEDIUM	semgrep	Detected possible user input going into a `path.join` or `path.resolve` function. This could possibly lead to a path traversal vulnerability, where the attacker can access arbitrary files stored in the file system. Instead, be sure to sanitize or validate user input first.	3	/apps/shell/ app.js:53
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	19	/spec/e2e/ nodesets/ amzn2023.yml

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.