

vets-website

github.com/department-of-veterans-affairs/vets-website

Scanned 29 July 2026 at 15:18 · Report generated 29 October 2026 · JavaScript · 276 stars

This repository is no longer available on GitHub. It was deleted or made private after this scan, so these findings cannot be re-verified. Read them as a record of what the repository contained on 29 July 2026.

OVERALL ASSESSMENT

Critical

Combined score 478 · 177 findings after filtering

Summary

This repository has a critical vulnerability score of 478.0, which means there are serious insecure coding patterns that an attacker could exploit to break into the system. The scan found many instances where code calls dangerous functions like `child_process` or uses shell commands with user input, but these issues appear to be ordinary mistakes rather than signs of malicious intent. While some findings involve using variables from GitHub workflows to inject code, which is a higher risk, the overall picture suggests the codebase has significant gaps that need immediate attention before it can be used in production.

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 (478)

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

MALICIOUS PATTERN

Safe (0)

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	53	104	20	177

83 of 177 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	Detected calls to <code>child_process</code> from a function argument <code>`appName`</code> . This could lead to a command injection if the input is user controllable. Try to avoid calls to <code>child_process</code> , and if it is needed ensure user input is correctly sanitized or sandboxed.	2	<code>/script/test-summary.js:129</code>
HIGH	semgrep	Detected calls to <code>child_process</code> from a function argument <code>`specPath`</code> . This could lead to a command injection if the input is user controllable. Try to avoid calls to <code>child_process</code> , and if it is needed ensure user input is correctly sanitized or sandboxed.	1	<code>/script/cypress-testrail-helper/lib/Utils.js:194</code>
HIGH	semgrep	Detected calls to <code>child_process</code> from a function argument <code>`portToKill`</code> . This could lead to a command injection if the input is user controllable. Try to avoid calls to <code>child_process</code> , and if it is needed ensure user input is correctly sanitized or sandboxed.	1	<code>/src/applications/_mock-form-ae-design-patterns/vadx/server/Utils/processes.js:23</code>
HIGH	semgrep	Detected calls to <code>child_process</code> from a function argument <code>`apiUrl`</code> . This could lead to a command injection if the input is user controllable. Try to avoid calls to <code>child_process</code> , and if it is needed ensure user input is correctly sanitized or sandboxed.	1	<code>/script/run-review-instance-api.js:105</code>

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	Bracket object notation with user input is present, this might allow an attacker to access all properties of the object and even it's prototype. Use literal values for object properties.	7	/src/applications/travel-pay/services/mocks/claims/claimHandlers.js:104
HIGH	semgrep	Found '\$SPAWN' 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.	2	/script/utils.js:25
HIGH	semgrep	Detected calls to child_process from a function argument `cmd`. This could lead to a command injection if the input is user controllable. Try to avoid calls to child_process, and if it is needed ensure user input is correctly sanitized or sandboxed.	2	/script/utils.js:25
HIGH	semgrep	Detected calls to child_process from a function argument `command`. This could lead to a command injection if the input is user controllable. Try to avoid calls to child_process, and if it is needed ensure user input is correctly sanitized or sandboxed.	2	/src/applications/_mock-form-ae-design-patterns/vadx/server/utils/processes.js:131
HIGH	semgrep	Using variable interpolation `\${{...}}` with `github` context data in a `run:` step could allow an attacker to inject their own code into the runner. This would allow them to steal secrets and code. `github` context data can have arbitrary user input and should be treated as untrusted. Instead, use	17	/.github/workflows/full-suite-tests.yml:255
HIGH	semgrep	By not specifying a USER, a program in the container may run as 'root'. This is a security hazard. If an attacker can control a process running as root, they may have control over the container. Ensure that the last USER in a Dockerfile is a USER other than 'root'.	2	/src/platform/utilities/preview-environment/Dockerfile:18

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	User controlled data in methods like <code>`innerHTML`</code> , <code>`outerHTML`</code> or <code>`document.write`</code> is an anti-pattern that can lead to XSS vulnerabilities	16	/src/applications/static-pages/widget-creators/createMyVAluginWidget.js:11
MEDIUM	semgrep	RegExp() called with a <code>`key`</code> 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/applications/vaos/utils/calendar.js:91
MEDIUM	semgrep	RegExp() called with a <code>`i`</code> 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 in	2	/src/platform/forms/validations.js:176
MEDIUM	semgrep	RegExp() called with a <code>`text`</code> 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/applications/benefits-optimization-aquia/shared/cypress-helpers/assertions.js:13
MEDIUM	semgrep	RegExp() called with a <code>`r`</code> 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 in	1	/src/platform/forms-system/src/js/routing/index.js:81

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `schema` 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	8	/src/applications/edu-benefits/10297/helpers.js:271
MEDIUM	semgrep	RegExp() called with a `labelText` 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-contr	7	/src/applications/benefits-optimization-aquia/shared/cypress-helpers/form-interactions.js:61
MEDIUM	semgrep	RegExp() called with a `props` 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-controlle	3	/src/applications/mhv-secure-messaging/components/MessageList/MessageListItem.jsx:43
MEDIUM	semgrep	RegExp() called with a `fileName` 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-contro	1	/script/github-actions/check-imports.js:13
MEDIUM	semgrep	RegExp() called with a `term` 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/applications/facility-locator/hooks/useServiceType.jsx:59

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	RegExp() called with a `rootUrl` 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-control	1	/src/applications/simple-forms-form-engine/shared/utils/url.js:5
MEDIUM	semgrep	RegExp() called with a `qs` 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 i	1	/src/platform/testing/e2e/mockapi.js:93
MEDIUM	semgrep	RegExp() called with a `domain` 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/platform/utilities/environment/stagingDomains.js:13
MEDIUM	semgrep	RegExp() called with a `batchRedirect` 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	/src/applications/proxy-rewrite/redirects/index.js:29
MEDIUM	semgrep	Service 'vets-website' is running with a writable root filesystem. This may allow malicious applications to download and run additional payloads, or modify container files. If an application inside a container has to save something temporarily consider using a tmpfs. Add 'read_only: true' to this se	3	/docker-compose.review.yml:3

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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 this data on the rendered page, consider placing it in the HTML portion (outside of	3	/src/platform/landing-pages/dev-template.ejs:282
MEDIUM	semgrep	Service 'vets-website' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-privileges:true' in 'security_opt' to prevent this.	3	/.devcontainer/docker-compose.yml:3
MEDIUM	semgrep	The application accepts potentially user-controlled input ``\$PROP`` which can control the location of the current window context. This can lead two types of vulnerabilities open-redirection and Cross-Site-Scripting (XSS) with JavaScript URIs. It is recommended to validate user-controllable input before	1	/src/applications/accredited-representative-portal/containers/AuthCallbackHandler.jsx:186
MEDIUM	semgrep	RegExp() called with a ``{ arrayPath, path }`` 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	1	/src/platform/testing/e2e/cypress/support/form-tester/utilities.js:24
MEDIUM	semgrep	RegExp() called with a ``field`` 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-controlle	1	/src/applications/caregivers/components/FormFields/AddressWithAutofill.jsx:115
MEDIUM	semgrep	Depending on the context, user control data in ``Object.assign`` can cause web response to include data that it should not have or can lead to a mass assignment vulnerability.	2	/src/applications/ivc-champva/10-7959C/config/submitTransformer.js:214

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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.	2	/script/run-cypress-enhanced.js:244
MEDIUM	semgrep	Detection of dangerouslySetInnerHTML from non-constant definition. This can inadvertently expose users to cross-site scripting (XSS) attacks if this comes from user-provided input. If you have to use dangerouslySetInnerHTML, consider using a sanitization library such as DOMPurify to sanitize your HT	2	/src/applications/avs/components/ParagraphBlock.jsx:19
MEDIUM	semgrep	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-exis	3	/src/applications/benefits-optimization-aquia/shared/utils/zod-helpers/zod-helpers.js:51
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.	54	/script/github-actions/annotate-enzyme.js:33
LOW	semgrep	A CSRF middleware was not detected in your express application. Ensure you are either using one such as `csrf` or `csrf` (see rule references) and/or you are properly doing CSRF validation in your routes with a token or cookies.	3	/src/applications/_mock-form-ae-design-patterns/vadx/server/index.js:9

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
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.	17	/script/test-summary.js:135

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 July 2026.