

kserve

github.com/kserve/kserve

Scanned 24 September 2026 at 05:53 · Report generated 29 October 2026 · Go · 5,992 stars

OVERALL ASSESSMENT

Critical

Combined score 878 · 1,104 findings after filtering

Summary

This repository has two distinct security issues that need immediate attention before it can be used in production. The most critical finding is a Cross-site scripting (XSS) vulnerability caused by untrusted user input, which means an attacker could potentially steal data or inject malicious code into web pages. Additionally, the code uses dangerous shell commands and processes files without proper validation, creating another layer of risk that attackers might exploit to run harmful scripts. While there are some signs of potential malicious intent, these appear to be standard insecure coding patterns rather than intentional attacks, so they can be addressed through routine code reviews. However, because the overall vulnerability score is extremely high at 864.0, the repository should not be deployed until these critical flaws are fixed by the development team.

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 (864)

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

MALICIOUS PATTERN

Low (14)

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	45	569	259	873

DETECTOR	LOOKS FOR	HIGH	MEDIUM	LOW	TOTAL
yara	Signatures of malicious behaviour in source files	26	0	137	163
bandit	Insecure coding patterns in Python source	2	18	42	62
guarddog	Malicious package and install-time behaviour	0	3	3	6

98 of 1,104 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 could be used to tamper with a web page rendering, which can lead to a Cross-site scripting (XSS) vulnerability. XSS vulnerabilities occur when untrusted input executes malicious JavaScript code, leading to issues such as account compromise and sensitive information leakage. To preve	2	/cmd/router/main.go:419
HIGH	bandit	tarfile.extractall used without any validation. Please check and discard dangerous members.	1	/python/storage/kserve_storage/kserve_storage.py:1713
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	31	/.github/workflows/automated-release.yml:34
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.	1	/hack/setup/scripts/generate-versions-from-gomod.py:54

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
HIGH	semgrep	Possible path traversal through <code>`tarfile.open(\$PATH).extractall()`</code> if the source tar is controlled by an attacker	1	/python/storage/kserve_storage/kserve_storage.py:1709
HIGH	bandit	subprocess call with shell=True identified, security issue.	1	/hack/setup/scripts/generate-versions-from-gomod.py:54
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'.	10	/localmodel.Dockerfile:39
HIGH	yara	YARA rule matched: detect_base64_exec — Encoded payload decoded and then executed	2	/python/custom_model/model.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	22	/hack/setup/scripts/generate-versions-from-gomod.py
HIGH	yara	YARA rule matched: detect_persistence_mechanism — Modifies a startup/scheduling location - legitimate application code essentially never does this	2	/python/huggingface_server_cpu.Dockerfile
MEDIUM	semgrep	Detected a python logger call with a potential hardcoded secret "Retrieved SP token credential for client_id: %s" being logged. This may lead to secret credentials being exposed. Make sure that the logger is not logging sensitive information.	1	/python/storage/kserve_storage/kserve_storage.py:1388

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	Semgrep detected a Kubernetes core API ClusterRole with excessive permissions. Attaching excessive permissions to a ClusterRole associated with the core namespace allows the V1 API to perform arbitrary actions on arbitrary resources attached to the cluster. Prefer explicit allowlists of verbs/resour	4	/hack/kserve_migration/kserve_migration_job.yaml:13
MEDIUM	semgrep	This pyproject.toml configures uv but does not set a dependency cooldown. Newly published packages can be malicious or unstable. Add `exclude-newer = "7 days"` under `[tool.uv]` to wait 7 days before resolving newly published package versions. Added in: 0.9.17 Reference: https://docs.astral.sh/uv/co	3	/python/huggingfaceserver/pyproject.toml:20
MEDIUM	semgrep	Detected a python logger call with a potential hardcoded secret "Created Secret: %s in namespace %s" being logged. This may lead to secret credentials being exposed. Make sure that the logger is not logging sensitive information.	1	/python/kserve/kserve/api/creds_utils.py:172
MEDIUM	semgrep	A secret is exposed in the workflow-level `env:` block, making it available to every job and step in this workflow — including any untrusted code run in pull-request workflows. Scope secrets as narrowly as possible: prefer step-level `env:` so the secret is only available where it is actually needed	4	/.github/workflows/e2e-test-quick-install.yaml:20
MEDIUM	guarddog	Detects access to sensitive files (credentials, configs, keys) [threat-filesystem-read]	3	/hack/setup/scripts/generate-images-sh.py:6
MEDIUM	semgrep	Detected a dynamic value being used with urllib. urllib supports 'file://' schemes, so a dynamic value controlled by a malicious actor may allow them to read arbitrary files. Audit uses of urllib calls to ensure user data cannot control the URLs, or consider using the 'requests' library instead.	5	/hack/setup/scripts/generate-versions-from-gomod.py:101

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	CORS policy allows any origin (using wildcard '*'). This is insecure and should be avoided.	1	/python/custom_model/model.py:145
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	3	/.github/actions/minikube-setup/action.yml:26
MEDIUM	semgrep	When working with web applications that involve rendering user-generated content, it's important to properly escape any HTML content to prevent Cross-Site Scripting (XSS) attacks. In Go, the `text/template` package does not automatically escape HTML content, which can leave your application vuln	5	/pkg/controller/v1beta1/inferenceservice/reconcilers/ingress/path.go:23
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'.	2	/hack/kserve_migration/kserve_migration.sh:96
MEDIUM	bandit	Audit url open for permitted schemes. Allowing use of file:/ or custom schemes is often unexpected.	5	/hack/verify-doc-links.py:227
MEDIUM	semgrep	Detected a possible denial-of-service via a zip bomb attack. By limiting the max bytes read, you can mitigate this attack. `io.CopyN()` can specify a size.	2	/pkg/agent/storage/https.go:223
MEDIUM	semgrep	Detected directly writing or similar in 'http.ResponseWriter.write()'. This bypasses HTML escaping that prevents cross-site scripting vulnerabilities. Instead, use the 'html/template' package and render data using 'template.Execute()'.	6	/cmd/router/main.go:411

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	bandit	Possible binding to all interfaces.	2	/python/ kserve/ kserve/ protocol/ rest/ server.py:136
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	8	/install/ v0.9.0/ kserve_kubeflow.yaml:15914
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	6	/install/ v0.5.0/ kf-serving_kubeflow_crds.yaml:6512
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	6	/install/ v0.12.0-rc0/ kserve.yaml:5233
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	6	/install/ v0.12.1/ kserve.yaml:927
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.10.1/ kserve.yaml:880

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.11.1/ kserve.yaml:9 12
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.11.0-rc1/ kserve.yaml:9 12
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.10.0-rc0/ kserve.yaml:8 80
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.11.0/ kserve_kubef low.yaml:906
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.11.0-rc1/ kserve_kubef low.yaml:906

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.9.0-rc0/ kserve_kubef low.yaml:1591 4
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.9.0-rc0/ kserve.yaml:1 5911
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/config/crd/ full/ serving.kserve. io_servin gruntimes.ya ml:3154
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.12.0-rc0/ kserve_kubef low.yaml:921
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.10.0-rc0/ kserve_kubef low.yaml:874

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.10.0/ kserve_kubef low.yaml:874
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.11.0-rc0/ kserve_kubef low.yaml:874
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.10.0-rc1/ kserve_kubef low.yaml:874
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.10.1/ kserve_kubef low.yaml:874
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	5	/install/ v0.10.0/ kserve.yaml:880

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	4	/install/ v0.12.0/ kserve.yaml:17444
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	4	/install/ v0.6.0/ kf-serving.yaml:6520
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	4	/install/ v0.14.0-rc0/ kserve.yaml:4603
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	4	/install/ v0.12.0-rc1/ kserve.yaml:16804
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	4	/install/ v0.5.0/ kf-serving_crds.yaml:14316

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.5.0/ kf-serving-crds.yaml:6520
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.5.1/ kf-serving-kubeflow.yaml:6516
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.12.0-rc0/ kserve.yaml:7878
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.8.0/ kserve-kubeflow.yaml:3199
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.5.0-rc1/ kf-serving.yaml:5837

SEVERITY	DETECTOR	FINDING	SEEN	EXAMPLE LOCATION
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.8.0-rc0/ kserve.yaml:3 202
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.7.0-rc0/ kserve.yaml:6 520
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.8.0/ kserve.yaml:3 202
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.6.0/ kf-serving_ku bepflow.yaml:6 514
MEDIUM	semgrep	In Kubernetes, each pod runs in its own isolated environment with its own set of security policies. However, certain container images may contain `setuid` or `setgid` binaries that could allow an attacker to perform privilege escalation and gain access to sensitive resources. To mitigate this risk,	3	/install/ v0.13.0-rc0/ kserve_kubef low.yaml:921

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.