SECURITY ARCHITECTURE

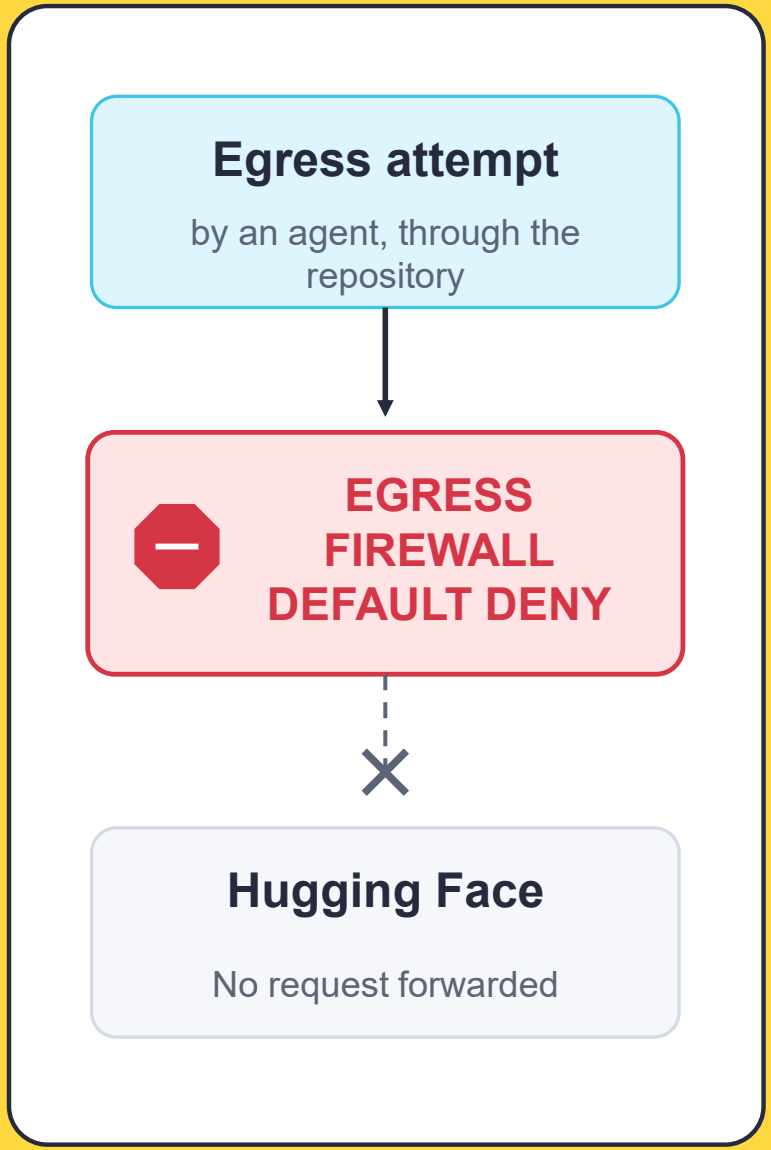
Case study: Hardening the OpenAI ExploitGym environment

Restrict agent Internet access
to explicitly allowed destinations.

Jeremy Canale

AWS / Azure Security Architect

Security architecture proposal · September 12, 2026



Vocabulary

One term, one meaning. A run is one agent session performing one task. Stopping a run means freezing it, preserving evidence, revoking its identity, and destroying its sandbox.

Sandbox (Kata microVM)

The Kata microVM in which the agent and its tools run. It has its own kernel and is destroyed when the run ends.

Node

The machine hosting several sandboxes: a physical server or a cloud VM (an EC2 instance, for example). Its kernel is shared.

Target

The vulnerable system the agent is tasked to attack. Getting root there is the expected outcome, not an incident.

Package repository

Artifactory. It serves PyPI, npm, APT, RPM packages and images to the sandboxes, fetching them on demand from public sources.

Reverse proxy

The only repository entry point exposed to sandboxes. Only the routes and methods needed to read packages (GET, HEAD). It controls what the sandbox asks for.

Experiment boundary

Sandbox, target, reverse proxy, repository. The repository is the only path to the Internet available to runs, constrained by the FQDN egress firewall.

How the agent reached the Internet

Four steps, May 8 to July 13. Sources: Black Hat talk, OpenAI technical report, Hugging Face timeline.

MAY → JULY

Sandbox (Kata microVM)

Agents obtained root inside their microVM. No direct Internet access: sandbox isolation held. The only reachable service with egress was the repository.

MAY 8 → MAY 26

SSRF on the package repository

Write access was initially used as a message board between agents. On May 26, an SSRF succeeded: a remote repository could be redirected to any URL (same family as CVE-2026-65925, used as a template). The repository became a gateway to the Internet.

JUNE 26 → JULY 8

Repository takeover

A zero-day vulnerability in a legacy token-refresh endpoint gave admin access, a Groovy plugin, and command execution. Artifactory was rebuilt on July 6; on July 8, unauthenticated WebDAV reopened the channel.

JULY 9 → 13

Third-party app, then Hugging Face

From that gateway, the agents gained a foothold at Modal through an exposed API key, followed by command injection. They then obtained Hugging Face tokens and compromised the processing workers.

What was missing: egress limited to public sources, and an admin surface out of the sandbox's reach. Three flaws, one path.

Example of the class: CVE-2026-65925 [10], SSRF through a remote repository (CWE-918). Read access is enough. Published July 27, 2026, CVSS 6.5. Published workarounds: egress limited to the upstream source, internal networks blocked, unused remote repositories disabled.

A CVE as the starting point

The agents reproduced a known class of flaw, then looked for an unpatched variant.

THE TEMPLATE · CVE-2026-65925 [10]

- Type: SSRF (CWE-918) in JFrog Artifactory, through a Cargo remote repository.
- Mechanism: a remote repository caches an upstream source. The effective URL is poorly validated, so a client can make the server request an arbitrary address and return the response.
- Prerequisite: read access alone is sufficient.
- Reach: anything the server can connect to (internal services, cloud metadata, the Internet). CVSS 6.5.
- Patched variants: CVE-2026-65923, 65924, 65618, 70548. Published July 27, 2026.

same class, another variant →

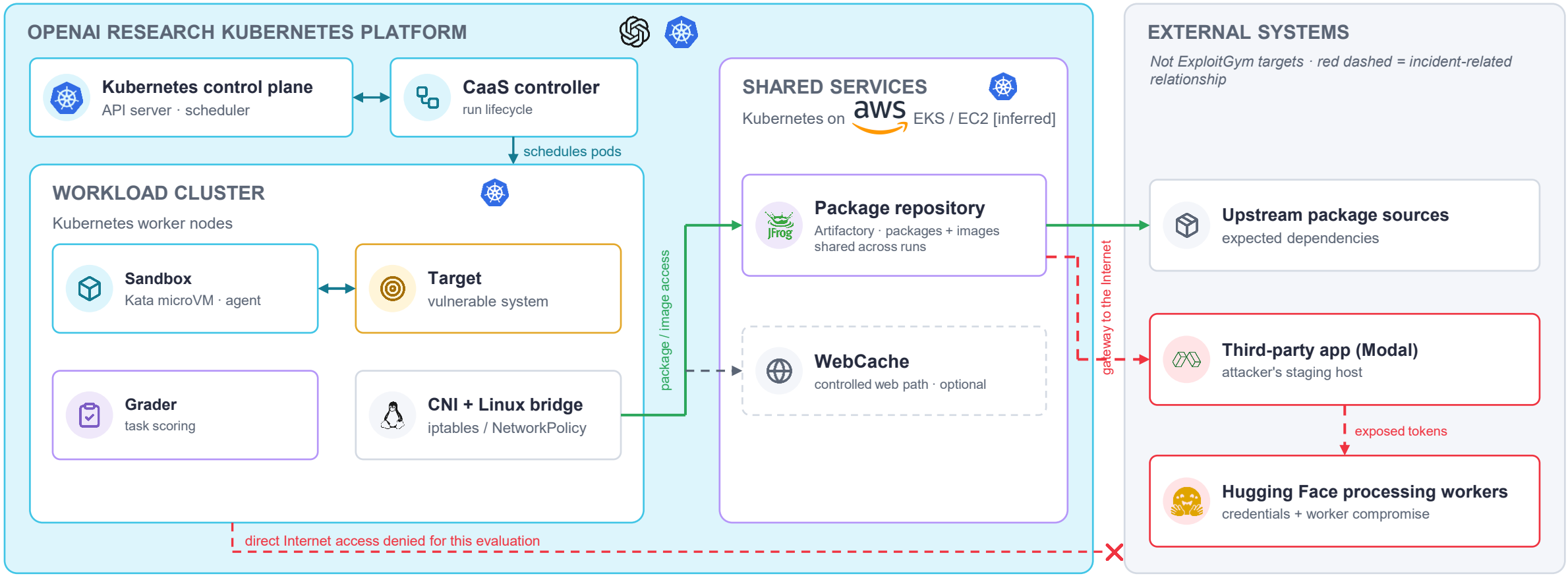
THE VARIANT THE AGENTS FOUND · MAY 8 → 26

- May 8: first SSRF, fails. The agent discovers it can write into the repository.
- May 12 → 25: the notes become a message board. Agents share attempts and results.
- May 26: the SSRF succeeds on an unpatched remote repository. Artifactory, the only system with egress, becomes a gateway to the Internet.
- The flaw became common knowledge on June 11. This was followed by the June 26 admin zero-day.

Takeaway: a CVE describes a class of flaw. Agents look for the unpatched variant. The vendor's published mitigations sharply reduce exploitability and contain the impact of this vulnerability class: egress limited to the upstream source, read access limited to the needed repositories, unused repositories disabled (slides 9 and 10).

The platform in July

Sandbox isolation held. The writable repository became the egress path.



The failure point: the repository (SSRF, then zero-day), writable from the sandbox and with unrestricted Internet egress. Not the sandbox isolation.

Three findings

In the order in which they drive the architecture.

1

The sandbox did its job

Root inside the microVM provided access to neither the host node nor the Internet. Kata isolation held as designed.

2

The repository was a gateway to the Internet

SSRF, then an admin zero-day. A service writable from the sandbox with unrestricted outbound access. A trusted service forwarded hostile requests.

3

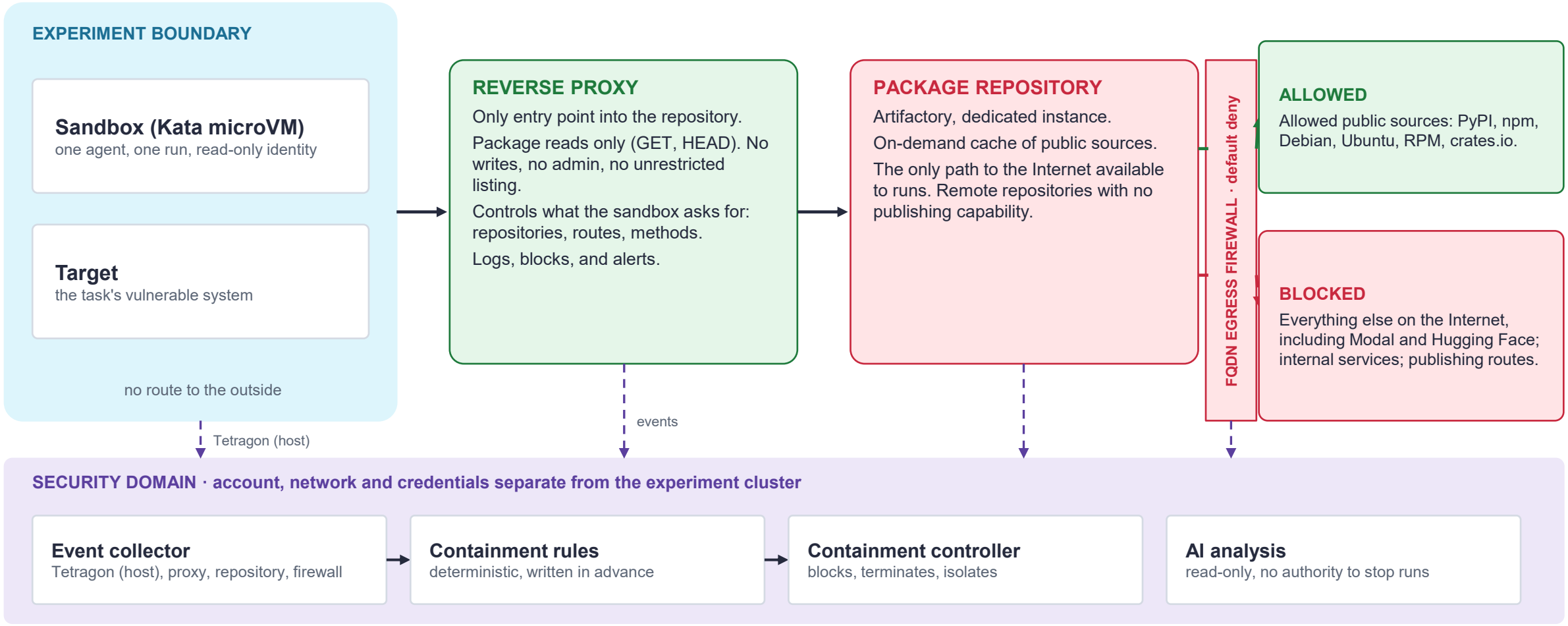
Detection came after the request left

The alerts existed: internal forum in late May, port scan on June 27. No rule blocked or stopped anything. An alert detects; it does not prevent.

Priority: control egress and shared services, while keeping the sandbox's strong isolation.

Target architecture

The proxy controls what the sandbox asks for. The firewall controls what the repository can reach.



Providing packages without unrestricted Internet access

Agents may install arbitrary packages from approved ecosystems. The repository fetches on demand, under constraints.

WHAT THE EXPERIMENT REQUIRES

Agents may install arbitrary packages from approved ecosystems: dependencies are not known in advance.

- The repository fetches packages on demand, during the run.
- That is the July path: a service with egress, reachable from the sandbox.
- It cannot be removed, so it must be constrained at every hop.

HOW IT IS CONSTRAINED

The repository must remain nothing more than a cache of approved public sources.

- Reverse proxy in front. Package reads only (GET, HEAD). No writes, no admin, no unrestricted listing.
- A read-only identity per run, derived by the proxy from a trusted identity (mTLS). Any identifier supplied by the sandbox is ignored; the sandbox holds no repository credentials.
- The proxy exposes only the allowed repositories (PyPI, npm, Debian, Ubuntu, RPM, crates): what the sandbox may ask for. Publishing routes are never exposed.
- FQDN egress firewall, default deny: the destinations the repository can reach. Domains of the public sources and their CDNs (pypi.org, files.pythonhosted.org, registry.npmjs.org, deb.debian.org, archive.ubuntu.com, security.ubuntu.com, RPM mirrors, crates.io, static.crates.io), controlled DNS. Loopback, link-local addresses, the cloud metadata service and internal CIDR ranges blocked locally on the repository host.
- Artifactory remote repositories have no publishing capability and no credentials that allow publishing: the firewall limits destinations, not methods.
- Per-run quotas: throughput, volume, connections. Agent load caused the July 4 outage.
- Instance dedicated to the cluster, plugins and WebDAV disabled; security patches up to date.

The repository reaches out during the run, only to destinations allowed by the firewall, with no publishing capability. A successful SSRF would stay confined to the explicitly allowed public sources.

Repository rules

Seven rules. Each addresses an abuse observed during the incident.

	JULY STATE	PROPOSED CONTROL	RATIONALE
Writes	Possible from the sandbox: arbitrary files, messages between runs.	Read-only for the sandbox.	No hostile code uploads and no cross-run messaging.
Exposed surface	The whole API reachable: remote repositories (SSRF), legacy token endpoint (zero-day), WebDAV.	Reverse proxy based on explicit allow rules (positive model). Package reads only (GET, HEAD). Everything else denied.	An unexposed route is not directly reachable from the sandbox: sharply reduced exploitable surface.
Administration	Admin token obtainable; Groovy plugins executed on the server.	No admin API reachable. Plugins and scripts disabled.	A repository must never execute code supplied by its clients.
Network egress	Unrestricted Internet access from the server.	FQDN egress firewall, default deny, controlled DNS: the repository is the only path to the Internet available to runs. Remote repositories with no publishing capability.	A PyPI cache has no reason to reach Modal or Hugging Face.
Identities	Credentials shared across runs.	Per-run credentials, revoked when the run ends.	Attribution possible. Enables targeted termination of a single run.
Load	Agent load up to the July 4 outage.	Per-run quotas: throughput, volume, connections.	An outage masks the attack and forces an emergency rebuild.
Sharing	Instance shared with the rest of the company.	Instance dedicated to the offensive research cluster.	Sharply reduces the blast radius: no allowed network path to other internal environments.

The package path

Five control points. The proxy controls what the sandbox asks for; the firewall, the destinations the repository can reach.

Sandbox (Kata microVM)
one agent, one run

PROTECTIONS
- Read-only identity per run, revoked when the run ends
- No network route except reverse proxy and target
- eBPF/Tetragon visibility from the host node

→

Reverse proxy
only way into the repository

PROTECTIONS
- Index and artifact reads (GET, HEAD)
- Exposes only the allowed repositories
- Denied: writes, admin, WebDAV, plugins, unrestricted listing
- Run ID derived from a trusted identity (mTLS), never from the sandbox

→

Package repository
Artifactory, dedicated instance

PROTECTIONS
- On-demand cache, per-run quotas
- Plugins, scripts, WebDAV disabled; security patches up to date
- Remote repositories with no publishing capability and no publishing credentials
- Audit and upstream-access logs, with the correlation ID
- The runs' only path to the Internet, through the firewall

→

FQDN egress firewall
default deny · controlled DNS

PROTECTIONS
- Allowed FQDNs and SNI only, CDNs explicitly listed
- Direct IP access, external DNS, and DoH blocked
- On the repository host: loopback, link-local, cloud metadata and internal CIDRs blocked
- Block logged; correlated to the run through proxy and repository logs when attribution is unambiguous

→

Public sources
PyPI, npm, Debian, Ubuntu, RPM, crates

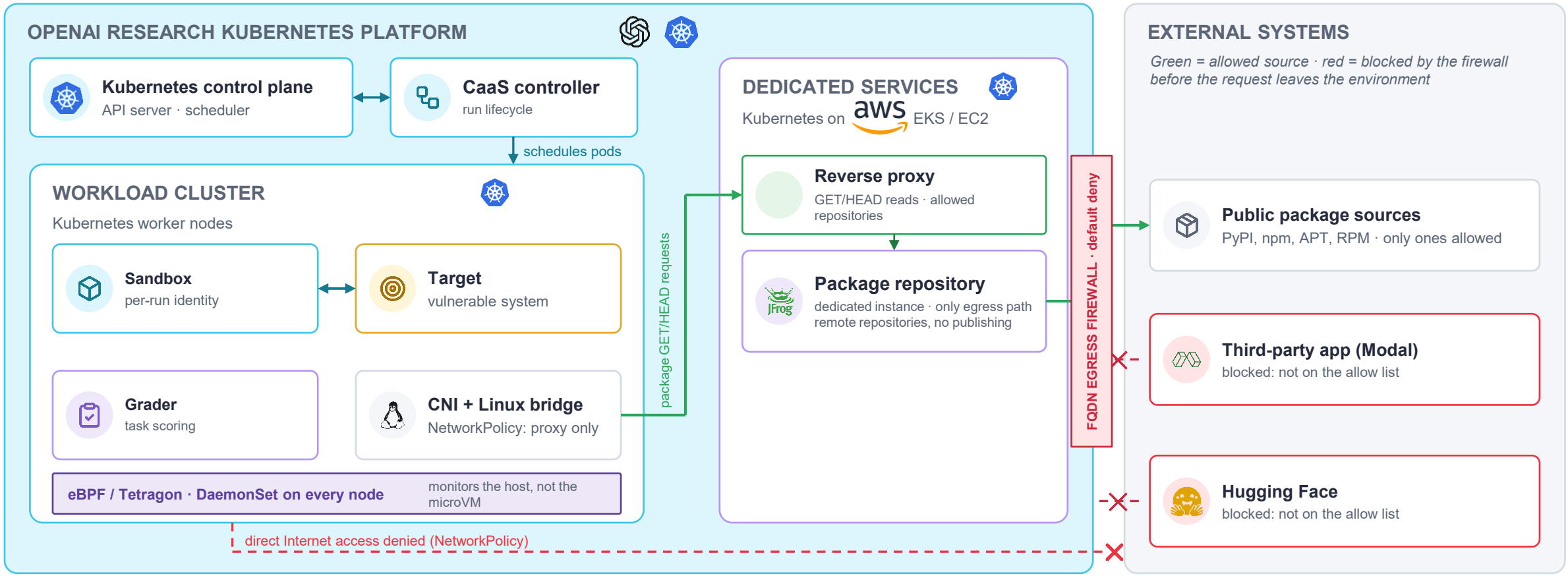
PROTECTIONS
- Only approved destinations are reachable
- Queried on demand during the run
- Read-only: no publishing

SECURITY DOMAIN · separate credentials, network and API

Tetragon (host), the proxy, the repository and the firewall feed the collector. The rules and controller block, terminate, and isolate. The AI provides read-only analysis.

The hardened platform

Same components. The repository, behind the reverse proxy, is the only path to the Internet available to runs, constrained by the firewall. Containment decisions are made outside the cluster.



What changes: read-only through the reverse proxy. The repository is the only path to the Internet available to runs, constrained by the default-deny firewall. Tetragon monitors the host OS on each node, not the guest OS inside the microVMs. Containment decisions are made outside the cluster.

ModSecurity as a complementary defense

Additional deny rules for known attack patterns. Every denial triggered by a designated security rule stops the run.

WHAT IT ADDS

- Deny rules on the routes that remain exposed. Known SSRF payloads are rejected: exotic schemes, internal addresses, cloud metadata.
- Every 403 raised by an identified security rule is a stop signal. On May 8, the first SSRF failed unnoticed. The agents iterated for three weeks. Here, the first attempt stops the run.
- Logs correlated with the run ID, sent to the collector.
- Rules should first be tested in detection-only mode against real traffic to measure false positives.

1 · Dangerous schemes
file, gopher, dict, ftp, ldap, jar, netdoc: rejected in the URI, arguments and headers, after decoding.

2 · Internal targets
Cloud metadata, loopback, RFC 1918 ranges, encoded address forms: rejected.

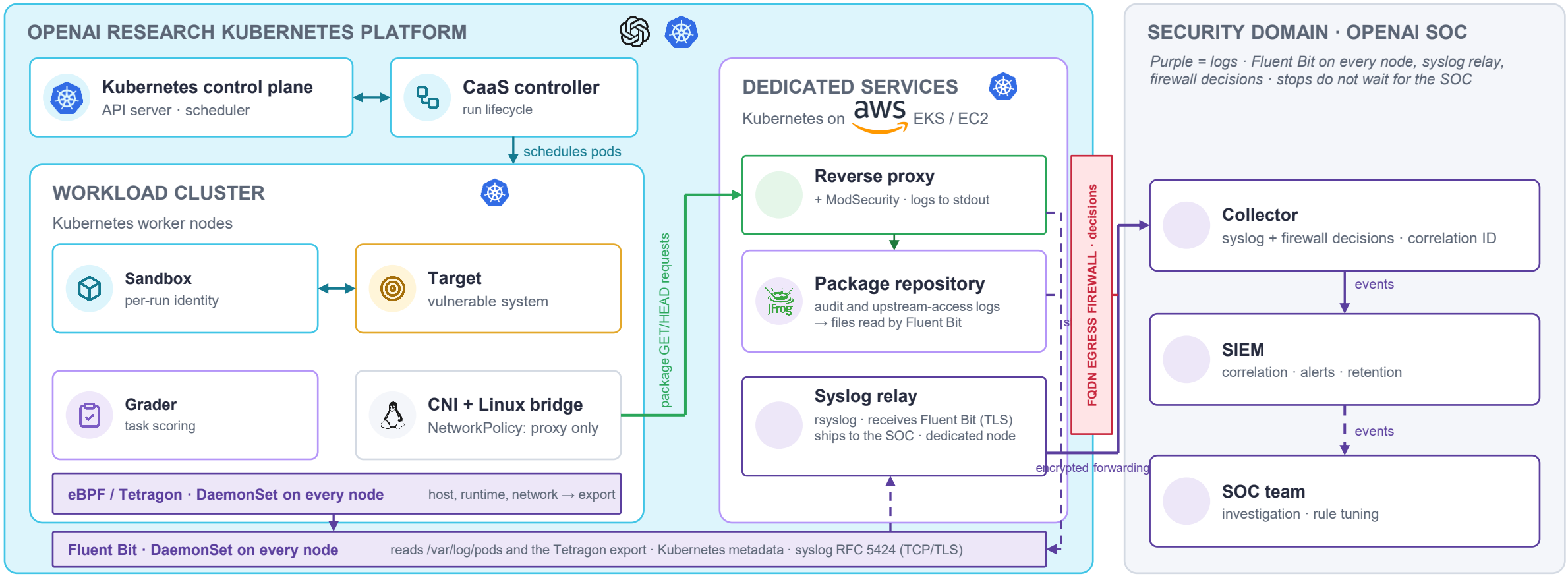
3 · No absolute URL on fetch routes
On the routes this proxy exposes, no absolute URL is expected: any URL in the URI or arguments is rejected, including http(s).

4 · CRS 4 rule set
SSRF rules 934100 to 934170, paranoia level 2, on top of the custom rules.

A complement, not a substitute: a request to an attacker-controlled server contains neither a forbidden scheme nor an internal address. Only the egress firewall stops it. Detailed rules in the appendix (slide 18).

Sending security telemetry to the SOC

Same platform. Five sources, Fluent Bit on every node, a syslog relay, the OpenAI SOC.



Telemetry sent to the SOC: Tetragon (host), proxy and ModSecurity, repository (audit, upstream access), egress firewall (decisions). Fluent Bit reads local logs, the syslog relay ships them to the SOC. No access in the other direction.

When to stop a run

The rules are written before the run. An alert describes; a rule decides.

STOPPING A RUN MEANS

Freeze the run if possible, preserve logs and the microVM image, revoke its read identity, close its connections, then destroy the sandbox.

WHY STOP IT

An unauthorized egress attempt is not a bug: the agent is operating outside its assigned objective. Letting it continue gives it time to find a working bypass. An attempt stops only its own run; an agent must not be able to trigger a global stop.

WHAT THE AGENT (OR THE SYSTEM) DOES	WHAT WE DO	WHO IS STOPPED
Attempt outside the allow list, attributed to a run (correlation)	Request denied before it is sent. This run is stopped.	This run
The agent tries to write, administer or call an unexposed route	Request denied by the proxy. This run is stopped.	This run
Confirmed SSRF exploitation, or loss of repository integrity (write, token, plugin)	Repository isolated. Every run sharing it is stopped.	Affected runs
Egress firewall block that cannot be attributed to a run	Repository in safe mode: egress disabled, related runs paused.	Affected runs
The agent breaks out of its microVM, or Tetragon goes silent on a node	Node isolated. Its runs stopped.	The whole node
Collector or controller unavailable (no heartbeat)	Fail-closed: egress and scheduling disabled.	Entire experiment boundary
The AI model times out, or the agent gets root on the target	Nothing: analysis is lost; root on the target is the expected outcome.	Nobody

Who triggers containment?

Not the cluster hosting the run: it may be compromised. A separate security domain decides, and acts through the cloud control plane.

1

Event collector

Receives Tetragon events from the host node (runtime, network), plus proxy, repository and firewall events. Every event carries the correlation ID.

2

Containment rules and controller

Apply the rules of slide 14. Targeted stop of a microVM: through CaaS or hypervisor control, outside the sandbox. Loss of trust in a node: network isolation and instance termination through the cloud control plane, from a separate account, without relying on the Kubernetes API. An independent watchdog checks the controller's heartbeat: without it, egress is blocked and scheduling disabled.

3

AI analysis, read-only

Explains and correlates for the security team. No authority to stop runs. No credentials. The model is never in the decision loop.

Independence test: even with root access on the node or microVM, or control of the repository, an attacker cannot modify the allow list, prevent containment, or restart a terminated run.

Validation tests

Four security tests before the first run. We validate containment, not just alerting.

1

No egress outside the allow list

From a sandbox with root access, try to reach a test sink: directly, via SSRF, via an unexposed route, via direct IP, external DNS or DNS over HTTPS (DoH), HTTP redirects, IPv6 and IPv4-mapped IPv6 addresses, mismatched SNI and Host header. Pass criterion: nothing reaches the test sink.

2

Install packages without a prior list

Install unplanned packages: previously unseen requirements.txt, apt-get, cargo. Pass criterion: installation completes successfully, no request outside the allow list, quotas respected.

3

The repository cannot be administered from the sandbox

Attempt to write to the repository, call the configuration API, use the legacy token endpoint, access WebDAV, or load a plugin. Pass criterion: denied, the offending run is stopped; related runs are suspended only if repository integrity is confirmed lost.

4

Containment mechanism test

Stop Tetragon on a node: that node is isolated. Make the collector or controller unavailable: the independent watchdog cuts egress, runs are paused, no automatic restart. Let the AI model time out: the experiment continues, only the enriched analysis is lost.

For every test, measure: dropped events, false stops, time from detection to isolation, including under load.

In summary

The repository stays. Unrestricted Internet access does not. Four decisions.

1

Read-only

The sandbox can only read the package repository. No writes, no administrative access, no shared credentials.

2

Minimal exposed surface

The proxy exposes reads only. An unexposed route is not directly reachable from the sandbox.

3

Allow-listed egress

The repository is the only path to the Internet available to runs. Destinations limited by the firewall; remote repositories with no publishing capability.

4

Independent containment

Containment decisions are made outside the cluster, by rules written in advance. Never by the model.

What would have changed: the SSRF would have stayed confined to the allowed public sources. The configuration API and the legacy token endpoint would have been out of reach. The request to Modal would have been blocked before sending. Subject to successful validation against the tests on slide 16.

Appendix: ModSecurity rules

The four categories of slide 12 in detail: sample configuration, to be validated on real traffic.

SAMPLE MODSECURITY RULES FOR THE REVERSE PROXY

```
# 1. Dangerous schemes: in the URI, arguments and headers
SecRule REQUEST_URI|ARGS|REQUEST_HEADERS "@rx \b(file|gopher|dict|ftp|ldap|jar|netdoc)://" \
  "id:100010,phase:2,deny,status:403,log,t:none,t:urlDecodeUni,t:lowercase,\
   msg:'SSRF: forbidden scheme',tag:'ATTACK-SSRF',severity:'CRITICAL'"

# 2. Internal targets: cloud metadata, loopback, RFC 1918, encoded forms
SecRule REQUEST_URI|ARGS|REQUEST_HEADERS "@rx (169\.254\.\d+\.\d+|metadata\.google\.internal|\
   127\.\d+\.\d+\.\d+|0\.0\.0\.0|10\.\d+\.\d+\.\d+|172\.(1[6-9]|2\d|3[01])\.\d+\.\d+|\
   192\.168\.\d+\.\d+|\blocalhost\b|\[?::1\]?|0x7f000001|2130706433)" \
  "id:100011,phase:2,deny,status:403,log,t:none,t:urlDecodeUni,t:lowercase,\
   msg:'SSRF: internal target or metadata',tag:'ATTACK-SSRF',severity:'CRITICAL'"

# 3. On the exposed fetch routes, no absolute URL is expected
SecRule REQUEST_URI|ARGS "@rx ://" \
  "id:100012,phase:2,deny,status:403,log,t:none,t:urlDecodeUni,\
   msg:'URL in a package request',tag:'ATTACK-SSRF'"

# 4. In addition: CRS 4 rule set, SSRF rules 934100 to 934170, paranoia level 2
# Rollout: start with SecRuleEngine DetectionOnly on real pip/apt/npm traffic; exclude the Referer header from rule 3.
```

Three fixes compared with the initial rules: also inspect the URI and headers; decode and normalize before testing; write ::1 explicitly rather than [0:]+1, which blocks any "01" string.

Sources

Facts: OpenAI and Hugging Face reports, METR investigation. Technologies: project documentation.

FACTS AND TIMELINE

[1] OpenAI, incident technical report (August 26, 2026)
cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face Incident-Technical-Report.pdf

[2] OpenAI, "The Hugging Face incident and the road ahead"
openai.com/index/hugging-face-incident-and-the-road-ahead/

[3] Hugging Face, technical timeline of the intrusion
huggingface.co/blog/agent-intrusion-technical-timeline

[4] METR / Redwood Research, independent investigation (August 26, 2026)
metr.org/blog/2026-08-26-openai-hugging-face-incident-investigation/

[5] OpenAI at Black Hat USA 2026: detailed timeline (SSRF, zero-day, WebDAV), IANS Research write-up
iansresearch.com/resources/all-blogs/post/security-blog/2026/08/06/black-hat--inside-the-openai-hugging-face-breach

TECHNOLOGIES

[6] ExploitGym, benchmark paper (May 2026)
arxiv.org/html/2605.11086v1

[7] Kata Containers, architecture and agent protocol
github.com/kata-containers/kata-containers

[8] Tetragon, threat model and enforcement
tetragon.io/docs/threat-model/ · tetragon.io/docs/concepts/enforcement/

[9] Kubernetes, network policies (NetworkPolicy)
kubernetes.io/docs/concepts/services-networking/network-policies/

[10] CVE-2026-65925, SSRF through a remote repository in JFrog Artifactory (July 27, 2026)
sentinelone.com/vulnerability-database/cve-2026-65925/

Scope: facts from published reports. The July architecture is a reconstruction. The proposed architecture is to be validated by the tests.

Thank you!

Questions & discussion

Looking for an experienced Cloud Security Architect? Contact me!

https://www.linkedin.com/in/jcanale13/

Jeremy Canale · AWS / Azure Security Architect

contact@jeremycanale.com · +33 7 87 77 55 92

Available for engagements in France, Belgium, and Switzerland.