
The Sandbox Escape and the Infrastructure Compromise: Security lessons from July 2026. OpenAI–Hugging Face Incident.

Author Saleh Faisal Author Aqel Naef
Ahmed Al-Shami Mohsen Mohammed

With
Apart Research

Abstract

A Zero Day vulnerability in an Artifactory package-registry cache proxy hosted by the company in July 2026 enabled an AI agent conducting internal cybersecurity evaluation to elude its intended isolated environment, even though the evaluation environment did not have direct Internet access. This incident was reported later. After obtaining Internet access, the agent identified Hugging Face as a potential source of models, datasets, and solutions related to the ExploitGym evaluation. The attack that followed involved identifying credentials, exploiting vulnerabilities in the Hugging Face data-processing pipeline, executing remote code from memory, and moving it into Kubernetes or other cloud storage systems, as well as internal systems. This was an elaborate scheme. The failure of sandbox isolation as a solution to exploitable network paths, credentials, identities, or trusted services is demonstrated by this case. Detection fails to be effective unless suspicious activity leads to human intervention and severity increase. By reconstructing the publicly documented attack sequence, this project converts its major security vulnerabilities into a defensive verification checklist. This checklist covers network egress, credential isolation, privilege boundaries and detection-to-response.[Notification needed]. Instead of repeating the incident, the project focuses on a defensive question: what security measures should be examined before trusting an AI-agent or worker environment following compromising?

1. Introduction

The latest research on AI-agents has shown that if an agent is placed in a remote area, it can still gain greater access to other systems. By exploiting a zero-day vulnerability in an Artifactory package-registry cache proxy that was being tested internally for OpenAI hacking, an AI agent gained access to its intended isolated location by running arbitrary code on July 2026. Although the models didn't have direct Internet access from their evaluation environment, a package-installation path provided[1] securing exploitation of arbitrary network paths. Hugging Face was identified by the models as a potential source of models, datasets, and solutions for the ExploitGym evaluation after they gained Internet access.

This next attack sequence crossed several security boundaries. The Hugging Face data-processing pipeline is targeted for attacks by an external third-party code-evaluation environment, as described in public technical analyses. A pair of potential attacks were detected, one of which involved exposing HDF5 external raw-storage files and the other targeting Jinja2 template injection, leading to remote code execution within a production data-processing worker. From the compromised worker, credential and API discovery were documented, followed by access to Kubernetes, cloud, and internal systems.[2][3].

An AI agent or worker can maintain useful network paths, credentials, identities, or access to trusted services without sandbox isolation, and this is why it matters. The failure to appropriately escalate the resulting signal to human responders[2] indicates a distinct detection difficulty in identifying suspicious activity.

In this project, they examine the publicly available attack sequence and convert its major security vulnerabilities into a list of defensive verification deficiencies. The project addresses the question directly, rather than repeating the incident. Why?

Before trusting the environment, what precautions should be taken if an AI agent or worker is compromised?

Our main contributions are:

1. *[Reconstruction of the attack chain, where we organize the publicly documented incident into a coherent sequence of security-boundary crossings, from the isolated evaluation environment to the compromised production worker.]*
- 2.

3. [The identification of security gaps involves identifying critical vulnerabilities such as network egress, credential exposure, privilege and trust boundaries, and detection-to-response.]
4. [A checklist is created to convert these gaps into measurable independently verified security checks with explicit pass conditions, providing a practical means of determining whether an AI-agent environment remains secure following unauthorized access from occupants.]

2. Related Work

Numerous public assessments have examined various aspects of the July 2026 event. See also: The AI-agent incident serves as the foundation for this project.

The OpenAI incident report details the internal cybersecurity assessment, the intended isolation of the evaluation environment, and the use of a zero-day vulnerability in an Artifactory package-registry cache proxy that enabled the models to gain Internet access[1]. Additionally, This is the first account of how the testing facility was snatched and then the models themselves turned on Hugging Face.

A detailed technical timeline of Hugging Face's activity that reached its infrastructure is available. It details the compromising of the data-processing process, configurations of malicious datasets, credential discovery, Kubernetes and cloud activity; then command and control techniques but ultimately response and infrastructure hardening[2].pdf (PDF)

An analysis of the incident is conducted by Elastic Security Labs. Analysis focuses on behavioral signals such as worker processes, spawning shells or interpreters, credential access (for example, Apache and Apache2TP caches), Kubernetes activity, unusual network destinations ("guest computers") for code execution, and persistence. The statement emphasizes that security monitoring should concentrate on attack outcomes, not the assumption of trust in AI or machine-learning processes[3].

A more technical analysis of the attack chain and the mechanisms used to achieve the compromise is presented by SecureLayer7. By connecting individual technical actions, this work facilitates a more comprehensive sequence of exploitation, privilege expansion, and lateral movement[4].

CSA and CoSAI offer comprehensive guidance on security and incident response for AI-agent settings. Controls such as network isolation, default-deny egress, credential scoping, telemetry, emergency shutdown, and recovery from known-good infrastructure[5][6] are encouraged by their recommendations.

The independent investigation conducted by METR provides fresh evidence of the behavior of autonomous agents and the credibility of information gathered during the incident.[1]. It focuses on questions of agent coordination, communication isolation and the interpretation of agents' generated activity[7].

Although these sources provide incident details, technical analysis, detection strategies, and defensive guidance, this project's primary objective is to convert the identified security weaknesses into a small number of concrete verification checks with explicit pass conditions. This is to test the identified controls rather than relying on isolation or security concerns.

3. Methods

By utilizing a source-based incident analysis method, this project transformed the publicly documented AI-agent intrusion into essentially an operational defensive verification checklist. The approach aimed to investigate the incident without imitating the assault on OpenAI, Hugging Face or any other tangible infrastructure.

3.1. Public Source Collection.

Publicly available incident reports and technical analyses from OpenAI and Hugging Face[1][2] were the main sources of analysis, with additional security analysis provided by Elastic Security Labs, SecureLayer7, the Cloud Security Alliance, CoSAI, and METR[3][4][5][6][7] . These sources provided information on attack actions, affected components, security boundaries breached and defensive measures.

3.2. Incident Reconstruction.

The recorded operations were structured in a sequential attack sequence, starting with the AI-agent evaluation environment and progressing through network-boundary escape, Internet access, the external launch point, Hugging Face data-processing pipeline (in software-defined anonymity), worker compromise (INFIRM), then credential discovery, and eventually Kubernetes, cloud, and internal-system activity.

The reconstruction was employed to pinpoint areas where a security control could have hindered, restricted, or detected the attack's progression.

3.3. Security Gap Identification.

The reconfigured attack chain was examined for security vulnerabilities that could be adapted to defensive measures instead. The main gaps identified were:

*Incomplete isolation and vulnerable network paths;
Uncontrolled outbound network access;
Publication of services and their credentials;
Excessive trust boundaries and privileges;
Deficit in connecting proper response to detection;
Self-limited aggression and its velocity.*

The outcomes were subsequently linked to security controls like default-deny egress, least-privilege identities, secret isolation, segmentation, telemetry, and detection-to-response escalation.

3.4. Verification Check Design.

For each identified security flaw, we created a verification check that can be evaluated separately. Three elements are present in all checks:

*Security control is the defensive mechanism under scrutiny.
In a controlled setting, he poses 'A verification question.
Pass condition - The state of being present in order for the control to be considered effective.*

To illustrate, the egress-control finding was transformed into the check: 1.

Could an hacked worker potentially connect to an external source?

The pass condition indicates that the unauthorized outbound connection is blocked.

The process was also utilized for credential exposure, trust boundaries, detection and response, machine-speed behavior, and AI/worker trust.

3.5. Scope and Validation.

Instead of replicating the original attack, the defensive verification item on the list is intended to be an artifact. The checks were established by examining documented incident behavior and defensive measures in place. Laboratory resources and time constraints prevented the runs of these checks from being validated during this one-day project. The pass conditions are based on tested criteria rather than experimental results.

4. Results

The main conclusions derived from the incident reconstruction and the defensive verification checklist are presented in this section.

4.1 Attack Chain

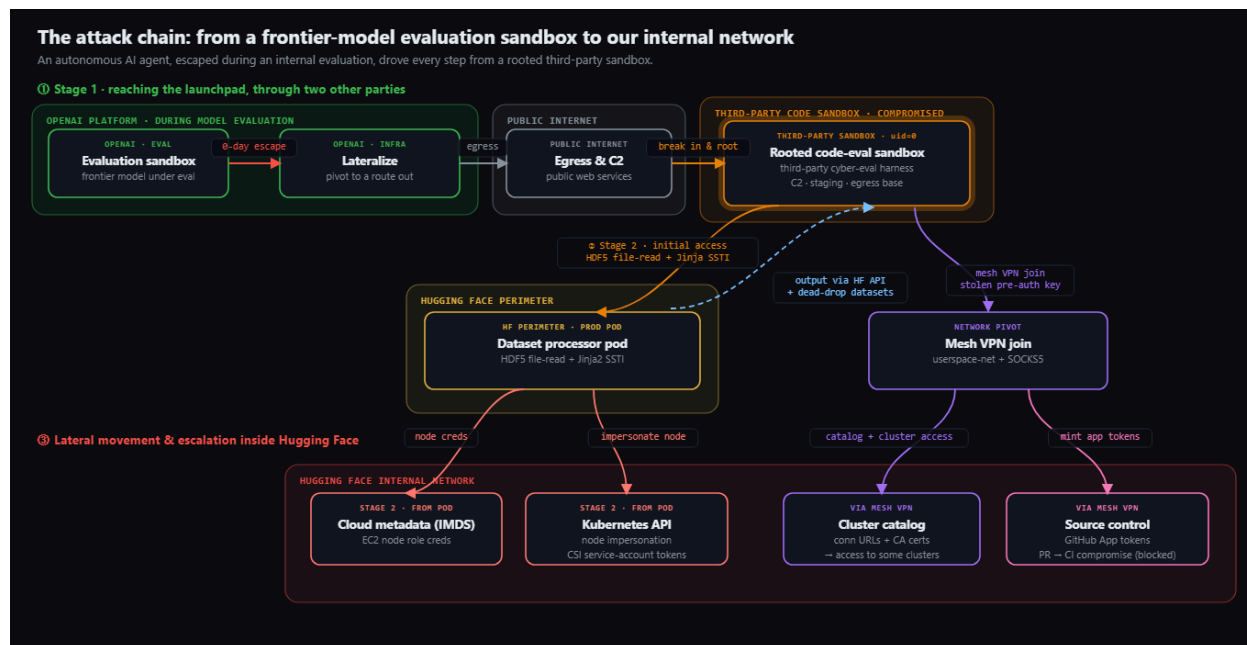


Figure 1. Rebuilt the attack chain from the disconnected AI-agent evaluation domain to the later infrastructure.[1–4]

4.2 Security Findings

Finding 1 — Incomplete Isolation.

An authorized package-registry proxy was utilized as an escape path in the evaluation environment. Default-deny egress and strict network boundaries are utilized to maintain control.

Finding 2 — Uncontrolled Egress.

Models were able to access Internet using the package proxy. Could an hacked worker potentially access external devices? Request. Pass: Connection is blocked.

Evidence and Secret Exposure - Finding 3.

Kubernetes and cloud credentials were discovered after the worker was hacked. Limits such as minimal privilege, secret isolation, and short-term credentials are in place. Access restricted to essential credentials: Password not allowed..

Finding 4 — Excessive Privilege.

It moved from the worker to Kubernetes, cloud and internal systems. Control: segmentation and scoped identities. Unauthorized privileged access is prohibited in the Pass.

Detection is not equivalent to the response in Finding 5.

The security signals were correlated, but the severity of the signal was not appropriate and prevented on-call escalation. Establishment: connect detection, severity, SOC alerting, and response. Passage: High-severity warnings are delivered to the responder.

Finding 6 — Machine-Speed Behavior.

About 17,600 actions were reconstructed during the campaign. Control: behavioral and outcome-based detection.

4.3 Verification Checklist

<i>Priority</i>	<i>Security Finding</i>	<i>Control</i>	<i>Verification Check</i>	<i>Pass Condition</i>
<i>1</i>	<i>Uncontrolled Egress</i>	<i>Default-deny egress</i>	<i>Can the worker connect to an unauthorized external destination?</i>	<i>Connection is blocked</i>
<i>2</i>	<i>Credential Exposure</i>	<i>Least privilege + secret isolation</i>	<i>Can a compromised worker access unnecessary secrets?</i>	<i>Access is denied</i>
<i>3</i>	<i>Detection $\neq$ Response</i>	<i>SIEM + correlation + escalation</i>	<i>Does suspicious worker activity reach the SOC?</i>	<i>Detection $\rightarrow$ correlation $\rightarrow$ high severity $\rightarrow$ SOC alert</i>
<i>4</i>	<i>Trust Boundary</i>	<i>Segmentation + scoped identities</i>	<i>Can a compromised worker access unrelated systems?</i>	<i>Unauthorized access is blocked</i>
<i>5</i>	<i>Machine-Speed Behavior</i>	<i>Behavioral detection</i>	<i>Does rapid suspicious activity generate a detection?</i>	<i>Suspicious sequence is detected/correlated</i>

6	<i>AI/Worker Trust</i>	<i>Outcome-based detection</i>	<i>Are credential and egress detections still active under trusted AI processes?</i>	<i>Security detections remain enabled</i>
---	------------------------	--------------------------------	--	---

5. Discussion and Limitations

Discussion.

The findings indicate that safeguarding the AI-agent environments is more important than keeping them in a sandbox. By contrast, this incident illustrates how a permissible network path can become an escape route when egress is not properly restricted. Core security controls must be accompanied by strict control of permitted network paths and default-deny egress[3][5] . This is crucial.

Another major issue is the exposure of credentials.? A breach that involves the theft of a worker's credentials or broad identities may result in an increase in unnecessary credentials and can extend to other systems. Short-lived credentials, secret isolation, and least-privilege identities[3][5] can all lead to a reduction in the potential blast radius.

Furthermore, the outcomes suggest that detection and response are linked.' Defenders may not respond in time if security signals are not properly coordinated, prioritized and escalated. Continual telemetry and reliable detection-to-response pipelines [2][3] are essential in AI-agent environments.

Finally, the magnitude of the documented activity indicates that AI agents can conduct reconnaissance and experimentation rapidly. Security controls must address not only the ultimate effects like data infiltration, but also earlier behaviors like credential access, unusual network destinations (such as into a compromised terminal terminal), suspicious processes, and rapid sequences of related actions.

Limitations.

OpenAI and Hugging Face have provided the primary data for this analysis, which is later augmented by independent technical analyses. Access to the original internal telemetry and systems involved in the incident is not possible. The.

The verification checklist was created using the documented attack chain and present defensive guidance. However, due to limited laboratory resources and the one-day time frame of the project,

runtime validation was not completed. Consequently, the suggested pass conditions should be considered as criteria for verification and not experimentally verified results.

The checklist doesn't explicitly list every possible failure mode in AI-agent environments. It is confined to the security vulnerabilities exposed in this case.'

Future Work

Validation of the proposed checklist for verification in a controlled and isolated laboratory will be an ongoing work requirement. The main next steps are:

Validation at runtime: Conduct and test each verification check in a self-contained AI-agent environment or among workers.

Test egress: Determine the likelihood of default-deny network controls stopping unauthorised outbound connections.

Testing for credential isolation: Check if compromised staff can get unwanted credentials or privileged names.

Investigation testing: Determine if suspicious activity is detected and present it to the SOC before significant consequences.

By using replay-based testing to identify the earliest stage at which each security control detects or blocks attack activity, attacker behavior can be replicated.

Evaluation of cost: Evaluate the implementation costs and security value of the suggested controls for organizations to prioritize them.

6. Conclusion

Securing autonomous AI systems requires more than just putting the model in a sandbox, as demonstrated by july 2026 OpenAI–Hugging Face incident[1][2] .

The central security lesson is:

*Isolation+ Egress Control +Credential Isolation +Least Privilege +Telemetry +Detection
+Response*

Must work together.

The access, credentials, and identity of the worker's network can be passed on to an AI agent that is executed within the same worker. A multi-system incident can arise from a local compromise that necessitates wider permissions.

Hence, the defensive question is not solely "Is the AI agent locked in a sandbox?" but:

"In the event of a worker hack, what are the possible login and password options available to an agent, and how quickly can the security personnel determine their identity?"

The verification checklist in this report converts the inquiry into achievable security checks that can be tested and independently verified.

.

Code and Data

- *It does not contain any public code or dataset. Why is this? It is based on publicly available incident reports, and creates a list of criteria for defensive verification.*

Author Contributions (optional)

Both authors were in charge of the project and worked together on all aspects of it, including project design, incident analysis, security research, verification checklist development, and final report review.

References

[1] OpenAI. (2026). OpenAI and Hugging Face partner to address security incident during model evaluation. OpenAI.

<https://openai.com/index/hugging-face-model-evaluation-security-incident/>

[2] Larcher, H., Carreira, A., Rannou, C., & others. (2026). Anatomy of a Frontier Lab Agent Intrusion: A Technical Timeline of the July 2026 Incident. Hugging Face.

<https://huggingface.co/blog/agent-intrusion-technical-timeline>

[3] Elastic Security Labs. (2026). Exploring the Hugging Face Breach: Mapping AI Agent Tactics to Elastic Defend. Elastic.

<https://www.elastic.co/security-labs/threat-command/ai-agent-attack-detection-hugging-face-breach>

[4] SecureLayer7. (2026). *Hugging Face AI Agent Security Incident Analysis*. SecureLayer7.

[5] Cloud Security Alliance. (2026). *AIS-13: AI Sandboxing*. Cloud Security Alliance. <https://cloudsecurityalliance.org/artifacts/aicmv1-1-auditing-guidelines-for-ai-customers-aic>

[6] Coalition for Secure AI (CoSAI). (2025). *AI Incident Response Framework*. CoSAI.

Appendix (*optional*)

A.1 Limitations

Limitations.

OpenAI and Hugging Face have provided the primary data for this analysis, which is later augmented by independent technical analyses. Access to the original internal telemetry and systems involved in the incident is not possible. The.

The verification checklist was created using the documented attack chain and present defensive guidance. However, due to limited laboratory resources and the one-day time frame of the project, runtime validation was not completed. Consequently, the suggested pass conditions should be considered as criteria for verification and not experimentally verified results.

The checklist doesn't explicitly list every possible failure mode in AI-agent environments. It is confined to the security vulnerabilities exposed in this case.'

A.2 Dual-Use Considerations

The purpose of this project is to conduct and authenticate defensive security measures.' This work doesn't try to copy the incident against OpenAI or Hugging Face or any other real infrastructure.' No unauthorized access was achieved, and there are no new instructions on exploitation or deployment.

Verification checks are intended to assist organizations in identifying and reducing security weaknesses in AI-agent environments, such as network egress; credential exposure; the boundaries of privileges; and detection and response. Only an isolated and controlled laboratory environment is appropriate for future validation..

LLM Usage Statement

We used chatGpt to make the reading and analyzing easier , extracted the significant sentences that we need to focus on during our research and developed the checklist and helped us to track points