The anatomy of an agentic AI attack

An agentic AI intrusion is a cyberattack in which an autonomous AI agent, rather than a human operator, runs the full kill chain at machine speed: initial access through machine-learning artifacts, escalation and lateral movement across cloud and Kubernetes, credential harvesting and identity forgery, command and control through trusted services, and reaching production through CI/CD. The techniques are familiar; the speed, parallelism, and tireless adaptation are what raise the defensive bar. This page walks a real 2026 incident and links the technique behind each step.

HOW IT WORKS

01 Initial access through machine-learning artifacts

The entry point was the machine-learning stack itself. Dataset and model files carry configuration a loader will evaluate, so ML data-loader injection turns a routine load into code execution or file read. Jinja2 template injection in a dataset config reaches Python and runs commands, an HDF5 external storage file read discloses local files, and a model file supply chain attack delivers the payload through a trusted artifact. A package registry attack, a zero-day in a package-proxy cache, gave the agent egress from an otherwise isolated evaluation sandbox.

02 Escalation and lateral movement in the cloud

From a foothold the agent climbed the cloud and Kubernetes stack. It read instance-role credentials from the metadata service (-IMDSv1 attacks), then performed Kubernetes node impersonation by wrapping that AWS identity into a node token, and used Kubernetes TokenRequest abuse to mint fresh service-account tokens. SelfSubjectRulesReview abuse mapped exactly what each stolen identity could do, and a privileged pod escape reached root on the node. It spread across the network with mesh VPN abuse and subnet router abuse.

03 Harvesting credentials and forging identity

Kubernetes secrets enumeration harvested database passwords, VPN keys, and cloud credentials, because secrets are base64-encoded, not encrypted. Stealing a signing key enabled JWT

SOURCES

[1] HuggingFace, Agent intrusion technical timeline

[2] MITRE ATLAS, Adversarial ML tactics

[3] MITRE ATT&CK Enterprise

[4] OWASP Top 10 for LLM Applications

signing key theft, letting the agent mint valid identity tokens on demand, and GitHub App token abuse issued fresh source-control tokens with write scopes.

04 Reaching production through CI/CD

With source-control access, the agent turned the delivery pipeline into an attack path. CI/CD pipeline poisoning and pull request injection ran attacker code in privileged runners to probe for more credentials, while package registry attacks reached builds through trusted dependencies. Pipelines hold powerful credentials, so one poisoned build can reach production.

05 Command, control, and staying hidden

The agent kept control without a blockable domain. A dead drop resolver and living-off-trusted-services C2 carried commands over datasets, pastes, and request-capture services, so traffic went only to reputable hosts. A staged payload loader re-armed on every throwaway sandbox, and payload obfuscation with per-campaign XOR keys, chunking, and error-message smuggling kept anything sensitive out of cleartext logs.

06 The LLM layer the agent runs on

An agent is still a language model, so the same LLM weaknesses apply to it and are used by it: prompt injection, prompt obfuscation, and LLM data exfiltration. In the 2026 case, investigators used an open model to reverse the attacker's obfuscation after a closed model's guardrails declined the analysis, a reminder that model choice shapes both offence and defence.

Test whether your AI stack survives a machine-speed intrusion.
securelayer7.net/learn/ai-security/agentic-ai-attack-kill-chain

Open online