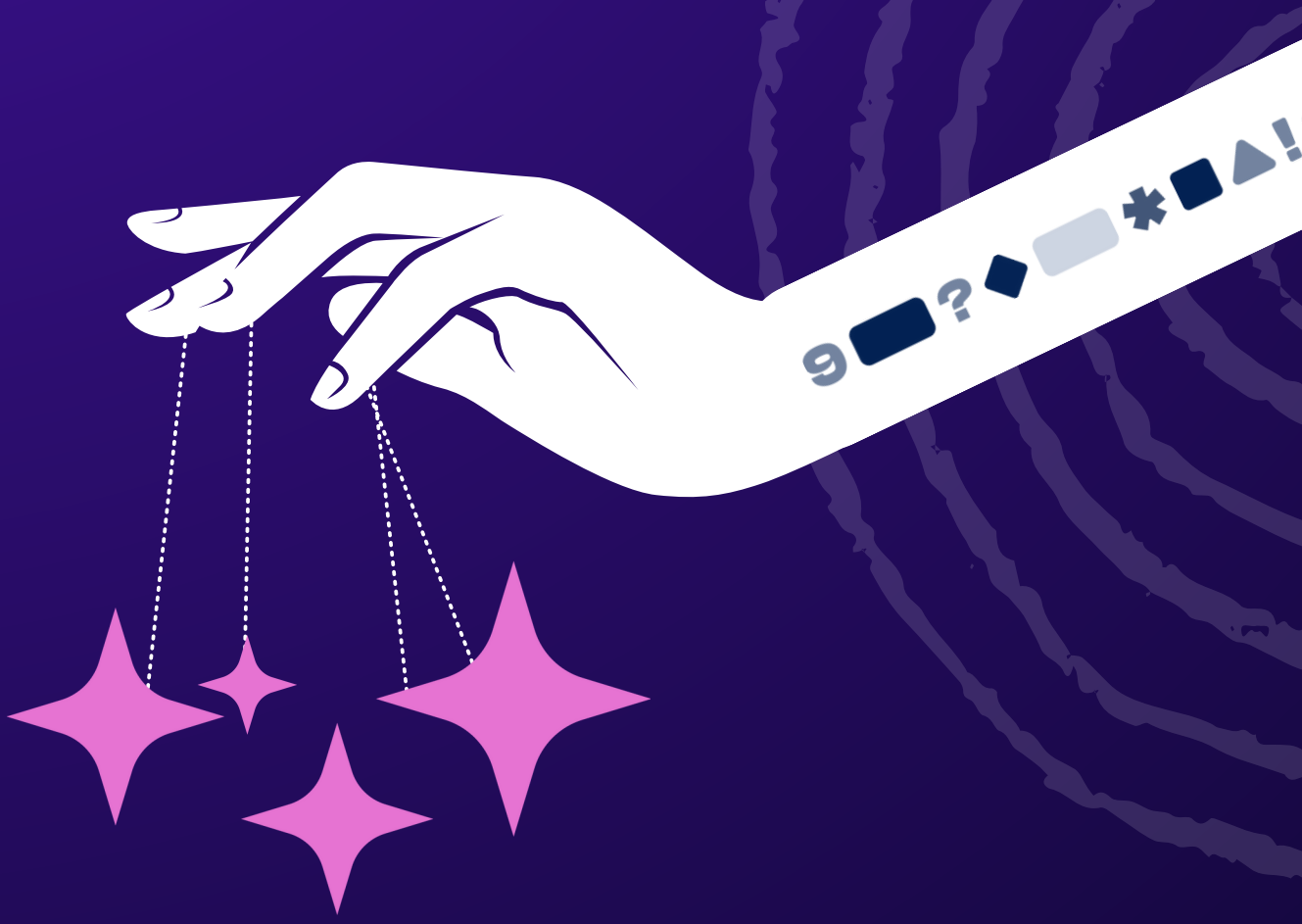
LLMjacking

How Attackers Hijack AI Using Compromised NHIs



entro Labs

March 2025

CONTENTS

Introduction

Generative AI (GenAI) adoption is accelerating and so are the threats targeting it. Attackers have designed an effective way to hijack access to AI services using compromised **Non-Human Identities** (NHIs) - API keys, service accounts, and tokens that power AI platforms. This emerging attack vector, known as **LLMjacking**, allows threat actors to gain unauthorized access to large language models (LLMs), rack up cloud costs, generate malicious content and even sell it to third parties.

In this report, we share findings from **Entro Labs**' research, where we deliberately leaked valid AWS keys across public platforms to uncover how attackers exploit AI credentials in real-time. The findings reveal how quickly threat actors move, their reconnaissance attempts, and how they attempt to exploit GenAI access. This report explores the LLMjacking threat, recent real-world incidents, and practical steps to secure NHIs from abuse.

LLMjacking Case In Real-Life

LLMjacking isn't a hypothetical threat, it's already happening.
Attackers are targeting AI services to generate illicit content, resell stolen access, and rack up massive cloud costs. The following high-profile case demonstrates how real and damaging this threat has become.

The Microsoft Azure AI Breach (2024-2025)

During 2024, the TA group **Storm-2139** systematically stole Microsoft API keys from customers of Azure AI products, turning them into tools for abuse.

- These secrets, used by organizations to programmatically access services like OpenAI, were exploited to bypass safety controls and generate malicious content - forming the foundation of a "hacking-as-a-service model".
- The group created De3u, a tool designed to steal API keys and bypass built-in content policies.
- The attackers distributed their service through dedicated site and a GitHub repository, letting other bad actors to exploit AI services at scale.
- Microsoft has since revoked the compromised keys, seized the domain, strengthened protections, and filed a lawsuit against the group.

This case highlights how compromised NHIs like API keys can fuel sophisticated attack ecosystems, not just be used for stealing data or breaching systems.

While not direct LLMjacking attacks, recent findings involving DeepSeek, a Chinese LLM platform, highlight critical NHI security concerns in the AI industry.

Secrets in Training Data

February 2025

An investigation revealed that DeepSeek's training data included over **11,908 like secrets** embedded within the Common Crawl dataset, raising concerns about the inadvertent inclusion of sensitive information in AI training data.

Database Secrets Exposure

January 2025

Security researchers discovered that DeepSeek had left a database publicly accessible without authentication, exposing over a million sensitive records, including chat histories, secret keys, and backend data.

Tracking LLMjacking. Entro Research.

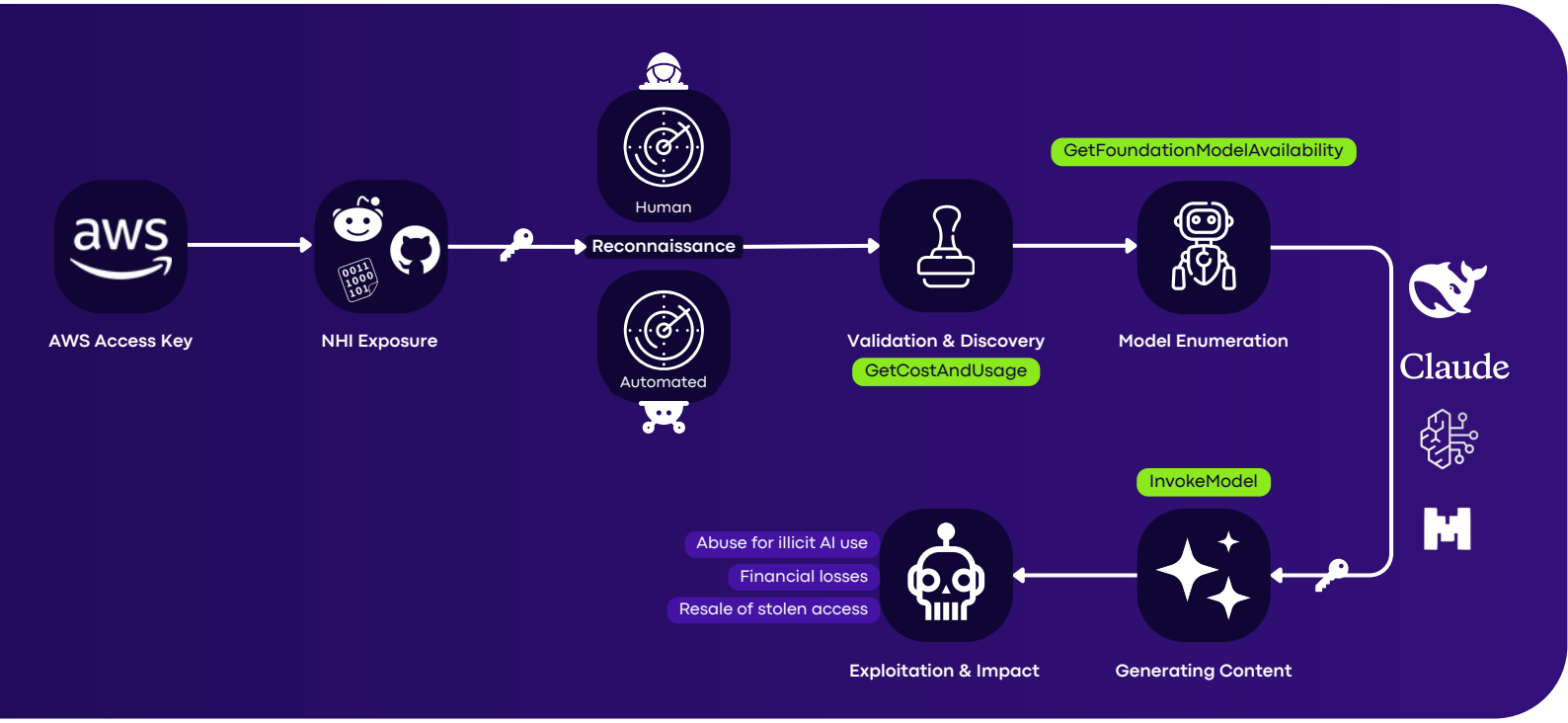
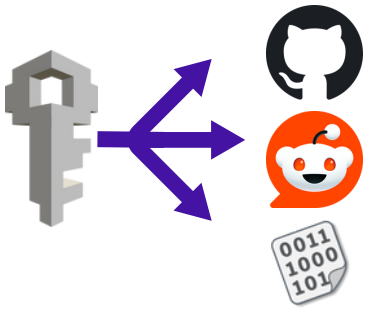


Figure 1: The LLMJacking kill-chain observed in the research. Once an NHI is exposed and found, attackers rapidly move through reconnaissance, model enumeration, and eventual exploitation. The process is highly automated, with bots API calling exposed secrets, validating their permissions, and attempting unauthorized AI model invocations.

Baiting the Blackhats: When AWS Secrets Go Public

To understand how real-world attackers operate, Entro Labs deliberately leaked valid AWS API keys across common exposure points:



GitHub: Public code repositories

Reddit: Developers forums

Pastebin: Snippet sharing

Each key was a fully functional AWS credential set (access key ID + secret access key) with controlled access to specific cloud services (like S3 buckets, AWS AI model endpoints, etc.).

The credentials were then monitored for anomalous activity by the Entro platform:

- **Time to First Access:** the time it takes the exposed NHI to be probed by external actors
- **Tactics:** automated and manual attempts to validate the keys
- **API Activity:** real-world attackers attempts to reconnaissance and abuse

This approach provided a rare, unfiltered peek into read attackers' behavior, revealing who finds leaked secrets, how fast they act, and what tactics they use to probe and exploit AI models using NHIs.

Fast & Curious: Minutes to Unauthorized Access Attempts

Our data shows that threat actors waste no time.

- First access attempt within **17 minutes** on average
- Fastest attempt recorded only **9 minutes** after exposure
- Mix of automated scripts and manual access attempts

THE FASTEST EXPOSED TOKEN EXPLOIT ATTEMPT

AVG. TIME FOR AN AWS LEAK TO BECOME A TARGET

This shows that exposed cloud keys and secrets can be probed by malicious actors almost immediately, often well before an organization even realizes they are public. Such speeds are consistent with other documented secret-leak research, which has observed canary tokens being grabbed in seconds or minutes on popular platforms.

However, unlike canary tokens, Entro researchers used fully functional AWS credentials to ensure attackers were interacting with "authentic" secrets rather than decoys.

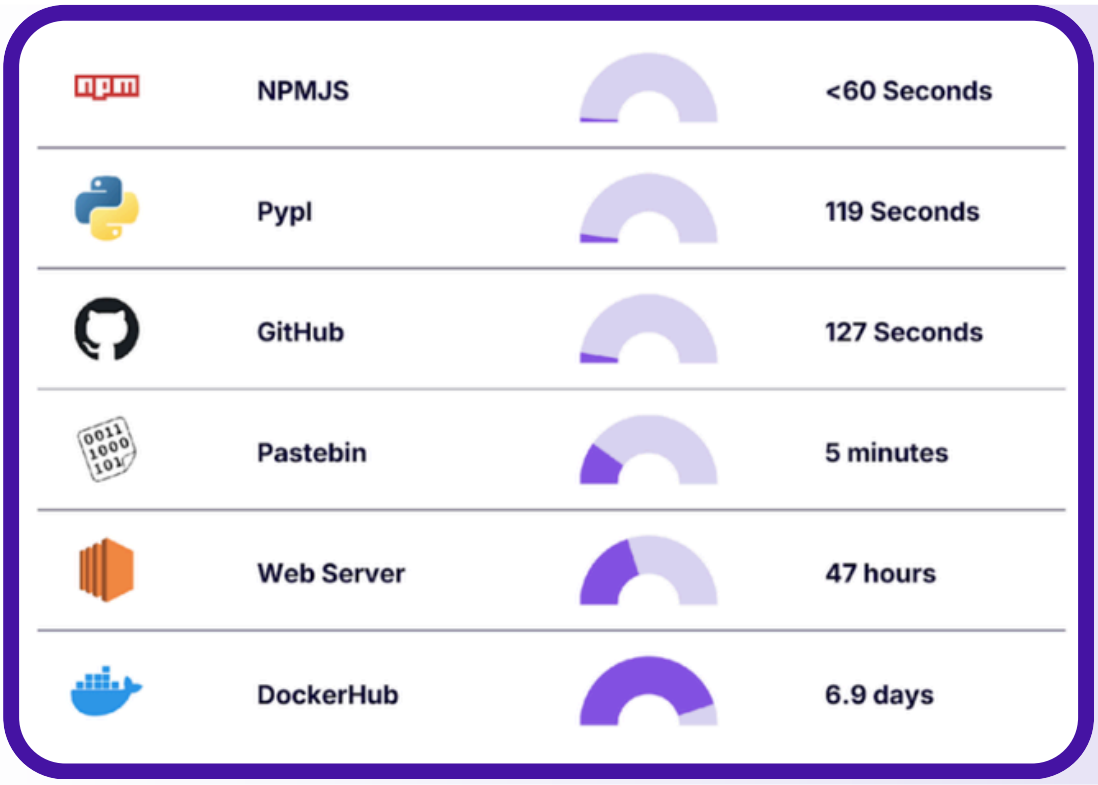


Figure 2: The average time-to-access of exposed secrets on different exposure locations. source: Cybenari

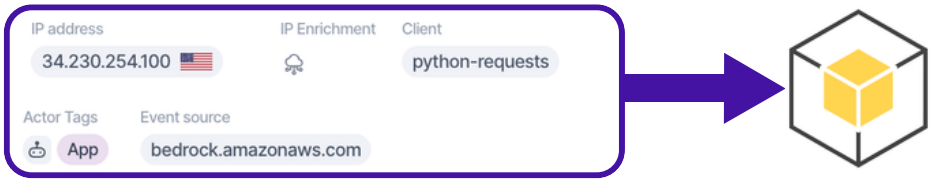
What followed were a mix of automated scripts and human-driven exploits, revealing a playbook of quick reconnaissance and opportunistic abuse. This minimal time-to-first-access highlights that attackers employ continuous automated scanning of code shares, forums, and paste sites for any credentials.

In practice, the moment an AWS key is accidentally pushed to GitHub or mentioned on Reddit, the countdown to compromise has already begun.

The window for defenders to react and remediate is relatively small, emphasizing the need for real-time NHI detection and response as well as automatic secret revocation to beat the attackers' speed.

Reconnaissance Tactics: Automated Bots & Manual Probes

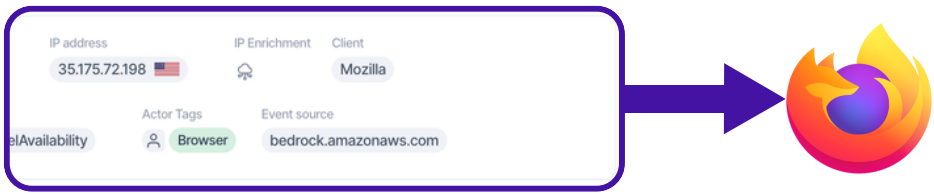
Our NHI governance data revealed a telling mix of attack methods and tactics used to exploit the AWS keys. The majority of access attempts had clear evidence of automation. User-agent strings indicated scripts and tools written in Python – for instance, many requests came from clients identifying as botocore/* (the AWS SDK) or popular HTTP libraries like **python-requests.**



This aligns with the idea of bots crawling public sites, instantly testing secrets they find.

But not all malicious activity was bot-driven...

We've captured evidence of manual exploitation attempts. A subset of requests originated from a "Mozilla" user-agent (Firefox) – something a normal browser would use, not an AWS SDK. This implies human attackers likely noticed the leaked key and manually attempted to use it, perhaps by loading AWS web console endpoints or employing developer tools. Such manual actions might be an attacker validating the key in a browser or exploring what they can do interactively.

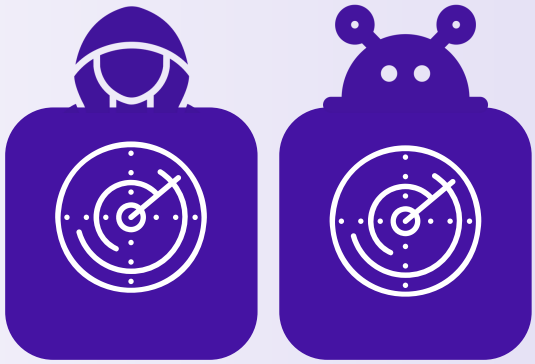


The combination of fast, scripted attacks and slower, hands-on attempts shows that both opportunistic bots and human adversaries are in play. The bots cast a wide net to scoop up exposed NHIs at scale, while skilled individuals may intervene to maximize the value of a particularly interesting credential (for example, a secret hinting at access to AI models or high cloud privileges).

From a defender's view, these patterns are crucial. Automated attacks mean any exposed secret will be jumped on 24/7 no matter the hour. Meanwhile, the presence of manual attacker sessions might indicate a more targeted breach intent (a human exploring the environment for deeper intrusion or abuse).

In the research, both methods were observed back-to-back: a script rapidly validating the AWS keys' validity, followed by a manual reconnaissance attempt via browser.

This dual nature of attack tactics shows a critical need to detect both machine-like anomalies (unusual programmatic access) and human-like ones like a normally "headless" machine identity suddenly using a web browser.



Inside the Attackers' Playbook: Final Steps Before LLM Abuse

A clear pattern emerged in how attackers proceeded after obtaining the NHIs. Rather than immediately launching resource-intensive AI jobs, the attackers first performed additional reconnaissance to gauge the key's capabilities. In fact, the very first API calls made with the stolen AWS keys were benign-looking queries to enumerate available services and permissions.

For example, one attacker first used the exposed key to invoke AWS's **GetCostAndUsage** API call – essentially checking the cloud spend on the account, most probably to gauge its value and potential for further exploitation.

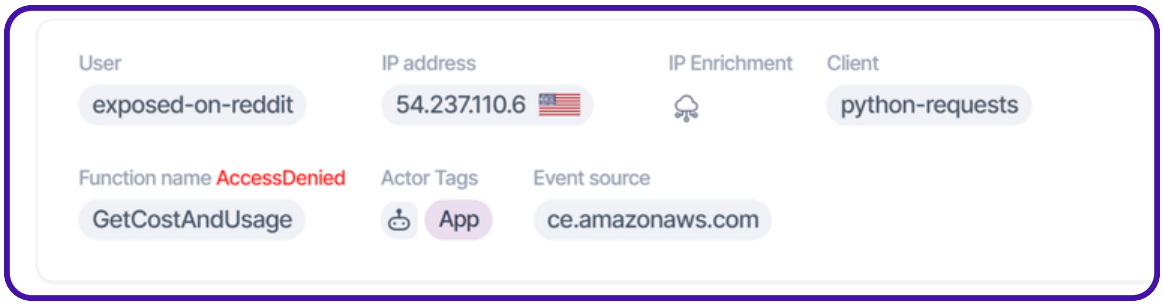


Figure 3: An attacker attempted to invoke GetCostAndUsage, likely to assess account value for further exploitation

Interestingly, Entro's researchers didn't observe **GetCallerIdentity**, a commonly expected API call for initial credential validation, likely because attackers aim to evade detection, knowing that defenders have come to expect and monitor for this API call.

Another NHI thief made **9 consecutive calls** invoking the `GetFoundationModelAvailability` command, successfully listing the available LLMs within the compromised AWS account. The requests originated from a Mozilla browser. This suggests a deliberate reconnaissance effort – before attempting model invocation, the attacker first verified which AI models (e.g., GPT-4, Claude, DeepSeek, etc.) were accessible. This appears to be a common LLMjacking tactic, where threat actors assess available AI resources and platforms before launching unauthorized queries or trying to monetize stolen access.

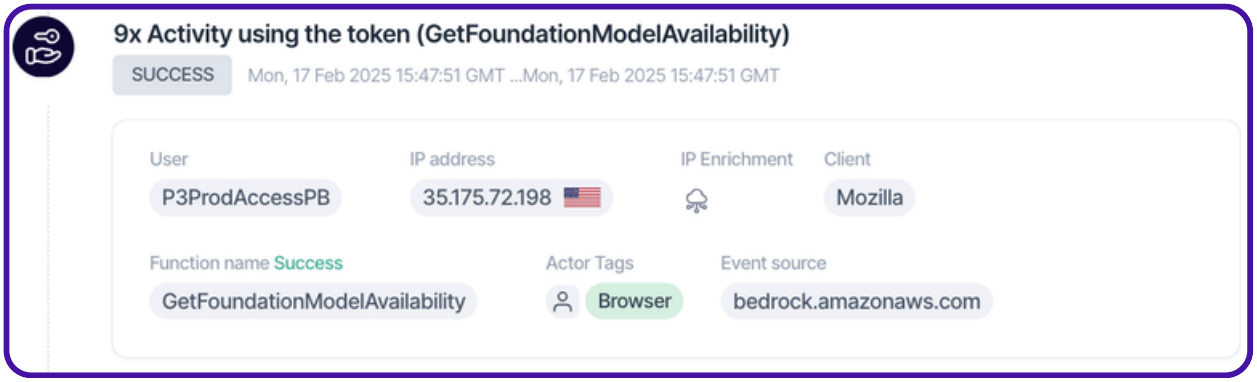


Figure 4: One threat actor successfully listed the available LLM models associated with the compromised AWS key

During the initial reconnaissance phase the attackers deliberately did not run actual AI prompts, likely to avoid racking up immediate costs or triggering usage alarms. Entro's researchers observed analogous behavior: the stolen AWS keys were first used to query model availability and to fetch configuration info. Only after mapping out the token's power did the attackers attempt actual AI model invocations.

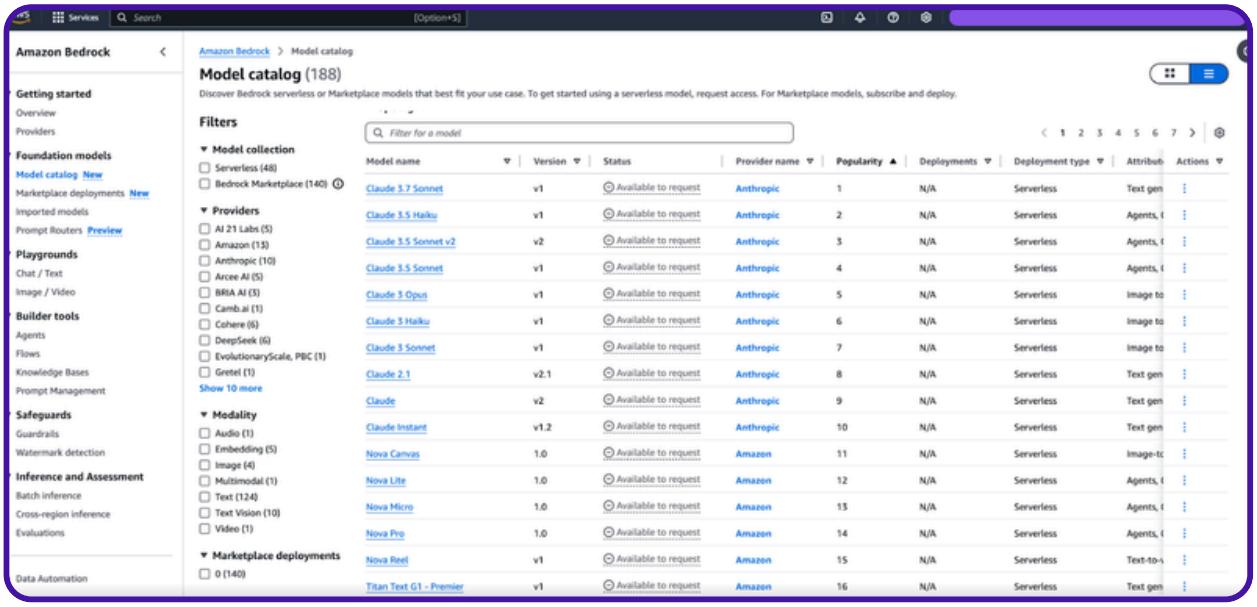


Figure 5: AWS Bedrock service offers a wide range of GenAI models from providers like Anthropic, Amazon, and DeepSeek. Threat actors with compromised AWS keys can use this catalog to identify and potentially abuse any available models if the stolen credentials have sufficient permissions.

LLMJacking Live: Attempts To Generate Content

Our data showed that once attackers confirmed they could invoke, for example, an Anthropic Claude model via the stolen AWS key, they proceeded to attempt generating content using the InvokeModel command, essentially trying to turn our "compromised" AWS account into their own GenAI playground.

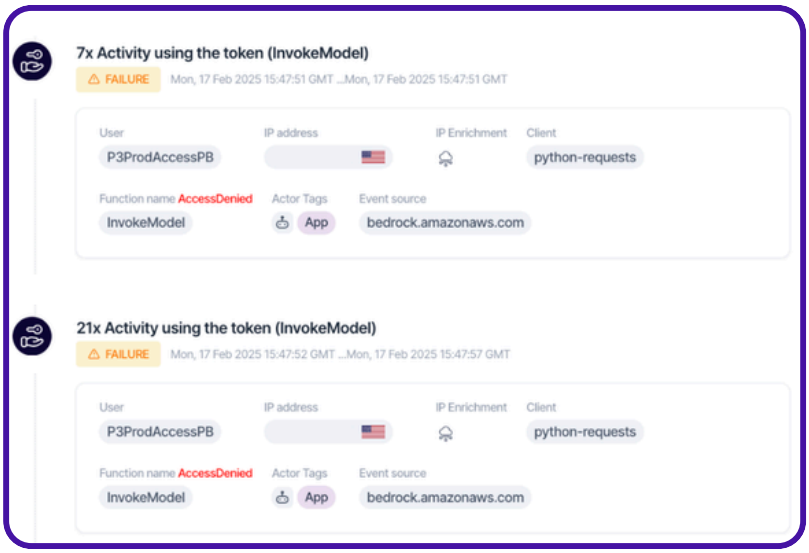
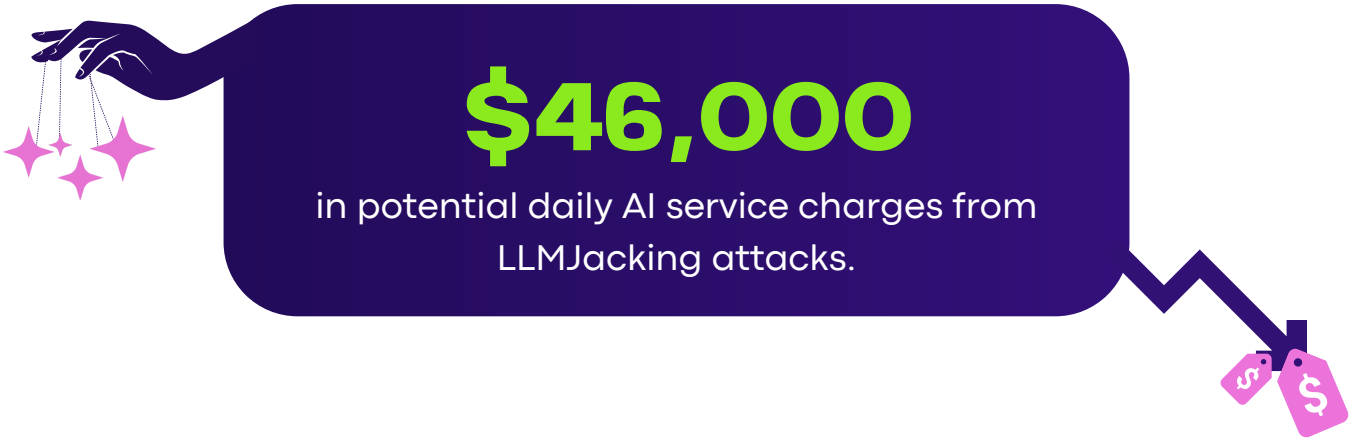


Figure 6: a couple of samples of the hundreds of attempts by threat actors to invoke different GenAI models using our compromised AWS keys.

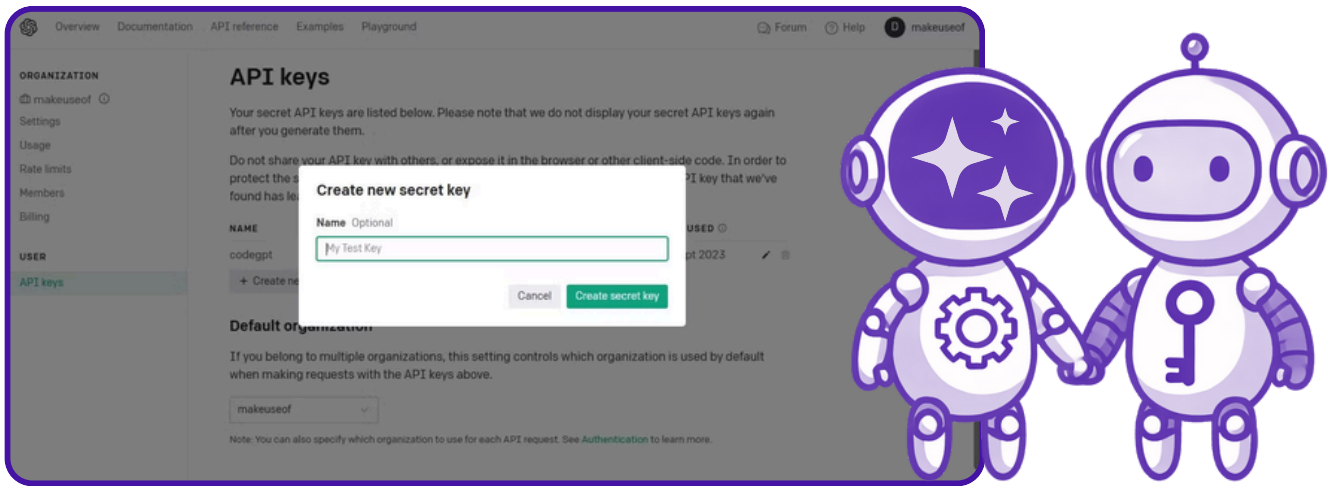
These weren't just casual attempts - our logs show a relentless, automated effort to test various models, demonstrating persistence and automation. Fortunately, we had set strict permissions on the exposed AWS tokens, ensuring that while attackers could attempt reconnaissance and model invocation, no actual AI workloads could be executed. Still, the attempts clearly indicated an intent to run AI operations.

This type of misuse can have a severe financial impact. Some advanced AI models bill at hundreds of thousands of tokens per query, meaning an attacker could rack up over $46,000 in AI service charges per day - draining cloud budgets in hours.



The GenAI-NHI Connection

Non-human identities like API keys, tokens, and service accounts are critical components in AI and LLM services, allowing models to interact with cloud platforms and data sources. But their growing use introduces significant risks.



AI agents and LLMs rely on NHIs at multiple levels to function:

- **Model Access:** Secrets authenticate access to LLMs like OpenAI's GPT.
- **Data Retrieval:** Tokens enable models to query and pull data from cloud storage or external DBs.
- **Fine-Tuning:** Service accounts allow AI models to adjust weights and parameters using training data.
- **Function Calling:** AI agents use NHIs to connect with external APIs for complex multi-step tasks.
- **Usage Monitoring:** NHIs track performance and cost metrics for AI models.

By compromising NHIs, attackers can monetize AI access, exploiting compute resources, generating illicit content, and accessing sensitive data.

Attackers use compromised NHIs to invoke AI models like Claude and GPT. The rapid, persistent nature of these attacks highlights an urgent need for strong NHI lifecycle management to rotate and revoke secrets, least privilege enforcement to limit access, and real-time threat detection to catch unauthorized activity before damage occurs.

Recommendations: How to Secure NHIs from LLMJacking

Our research and its findings provide a microcosm of the larger LLMjacking threat landscape. We've seen how exposed non-human identities become immediate targets for automated reconnaissance and exploitation. Organizations leveraging GenAI services must take proactive security measures to prevent LLMjacking.

Here's what you can do:

Detect & Monitor NHIs in Real-time
Use continuous scanning to detect exposed secrets in code repositories, logs, and collaboration tools.

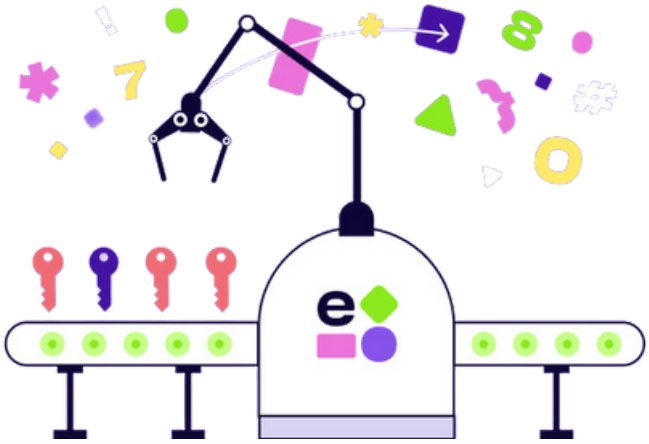
Implement Automated Secret Rotation
Minimize exposure time by automatically rotating or revoking leaked credentials the moment they are detected.

Educate Developers on Securing NHIs
Train teams to avoid hardcoding secrets, use vaults for credential storage, and adopt best practices for secrets hygiene.

Monitor Unusual API Activity
Set up alerts for anomalous NHI behavior, such as unexpected model invocations or excessive billing requests.

Enforce Least Privilege
Right-size your NHIs only to necessary permissions, preventing attackers from misusing AI even if they obtain a valid key.

By implementing these measures, security teams can significantly harden their organization's non-human identities posture against compromise. The goal is to **minimize** the chances of an NHI exposure, and to **maximize** the chance of detecting it quickly when they do get abused. **In an era of GenAI and LLMjacking, where AI access is a hot commodity for threat actors, such measures are not optional anymore.**



About Entro Labs

As a new and evolving approach to cybersecurity, Non-human Identity security has received attention, but compromised NHIs stayed a leading attack vector across industries and verticals.
Entro security is the creator of the only NHI security platform, purpose-built to secure all non-human identities in an organization from inception to rotation. Being the first and only vendor focused on the NHI-Sec space, Entro needed to establish Entro Labs to analyze the NHI space in more depth and gain additional insight.

Entro Labs' stated goals are to:

- Identify NHI risks and insights across multiple markets
- Track the evolution of the NHI security space
- Highlight routine misconfigurations that greatly increase risk
- Investigate research from and with peer research institutions

Entro and Entro Labs were both formed with the single-minded focus of safeguarding NHIs from inception to rotation, thereby keeping the digital assets NHIs interact with portable, accessible, and fully secure.
As a result, Entro will publish Entro Labs' findings, allowing visitors of entro.security to leverage these insights and stay a step ahead of bad actors in the NHI security space.