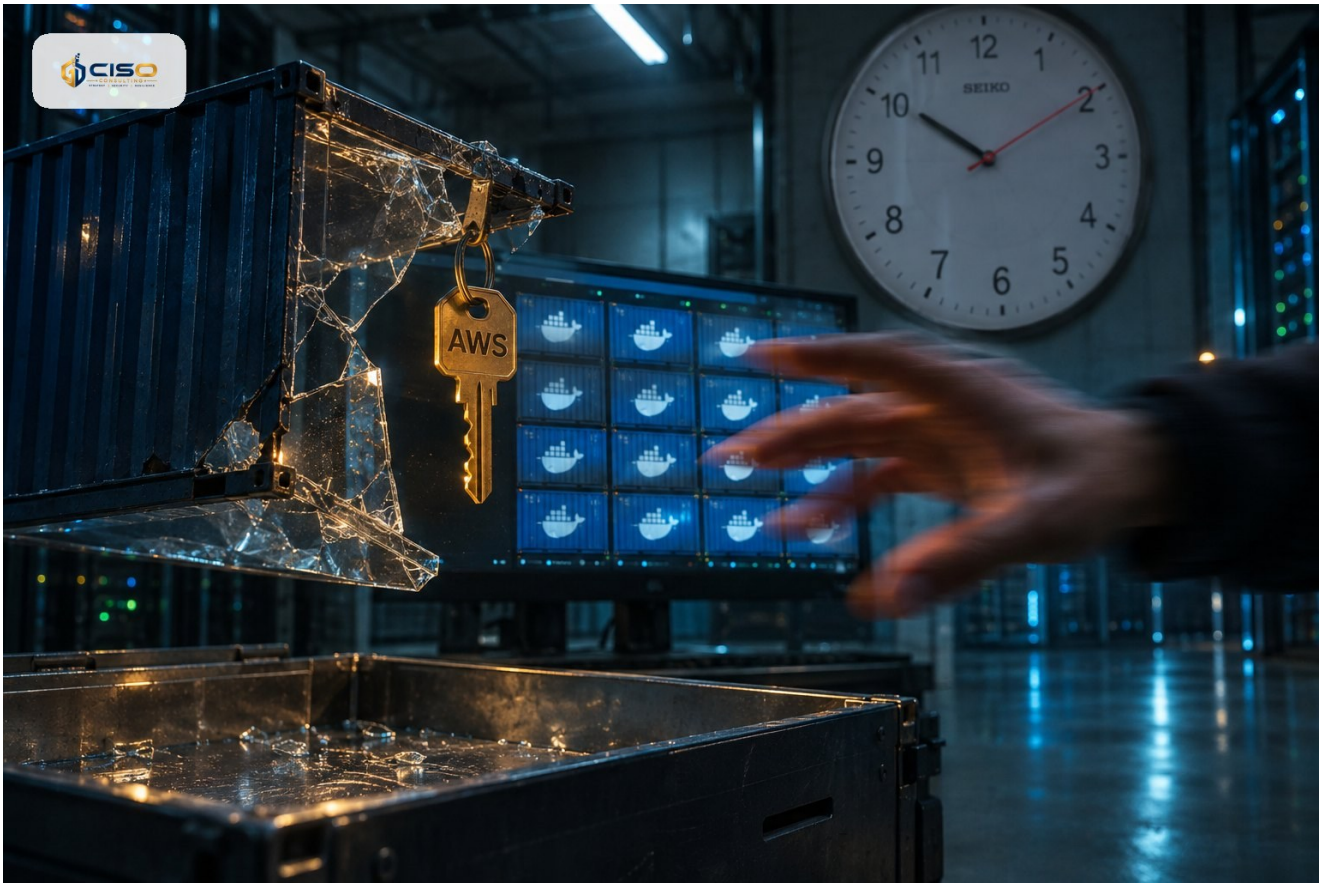
Leaked cloud keys in public Docker images were used within seconds, research finds



Stony Brook researchers planted decoy credentials in 300 public Docker images. Bots used the AWS keys within seconds, so a leaked key should be treated as compromised the moment it is published.

- Decoy AWS, OpenAI and Anthropic keys were placed in 300 copies of the nginx image across eight registries over four months.
- The AWS keys received over 140,000 API calls from more than 5,000 IP addresses; the fastest first call came 7.86 seconds after publication.
- Median time to first use was 51 seconds, and every AWS key was used within 15 minutes.
- OpenAI and Claude keys were drained of their full balances; restricted-permission keys were never used to consume tokens.

> - Images with names suggesting sensitive infrastructure drew more than double the pulls on Docker Hub.

Researchers Billy Tsouvalas and Nick Nikiforakis of Stony Brook University published 300 copies of the official nginx image on eight public registries, including Docker Hub, Amazon ECR and Red Hat Quay. Each held decoy AWS, OpenAI and Anthropic keys. Nothing was advertised, yet the AWS keys received more than 140,000 API calls from over 5,000 IP addresses.

What the experiment showed

In the first round, on Docker Hub, more than half of the AWS keys were used within a minute and all within 15 minutes. The median time to first use was 51 seconds. The OpenAI and Claude keys were emptied of their balances, in some cases after a waiting period of weeks, which the authors speculate reflects validated keys being resold.

The bots appear not to run the images at all. They watch a Docker Hub catalog endpoint listing new images, read each manifest, download only the layers likely to hold credentials and search them for key-like strings. Docker Hub's pull counter also lags real downloads by about 5 to 6 minutes, so an owner watching it cannot react in time.

Implications

The practical lesson is that publication and compromise are effectively the same event. Keys with restricted permissions were not used to consume tokens, which supports least-privilege scoping. The researchers suggest registries scan pushed images for known key formats and restrict the public list of newest images. Docker had acknowledged the disclosure and was reviewing it when the paper was written.

For institutions that build and publish container images, the focus should be on preventing secrets from entering images and on fast revocation if one does.

What to do now

1. Treat any credential found in a published image as compromised and revoke it immediately; do not rely on pull counts to judge exposure.
2. Add secret scanning to CI/CD pipelines so images are checked for keys and login details before they are pushed to any registry.
3. Issue cloud and AI API keys with least-privilege, restricted permissions and set spending limits and usage alerts.
4. Review which container images your teams publish publicly, and prefer private registries for internal workloads.
5. Rotate cloud and AI keys regularly and monitor AWS activity logs for unexpected identity checks or calls

from unknown IP addresses.