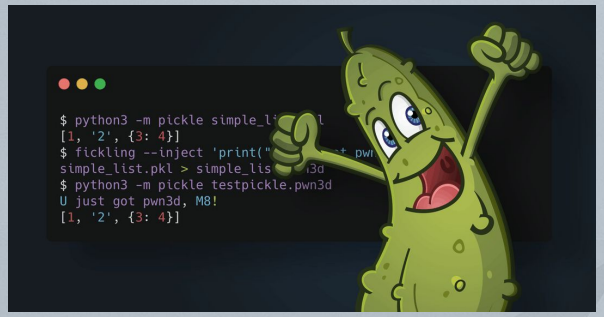


TRAIL OF BITS

Never a dill moment

Exploiting machine learning pickle files

What is Pickling?

Pickle: module for serializing and de-serializing arbitrary Python objects

Pickling: process of converting Python object into byte stream

Unpickling: converting a byte stream back into a Python object

Allows you to serialize custom objects easily

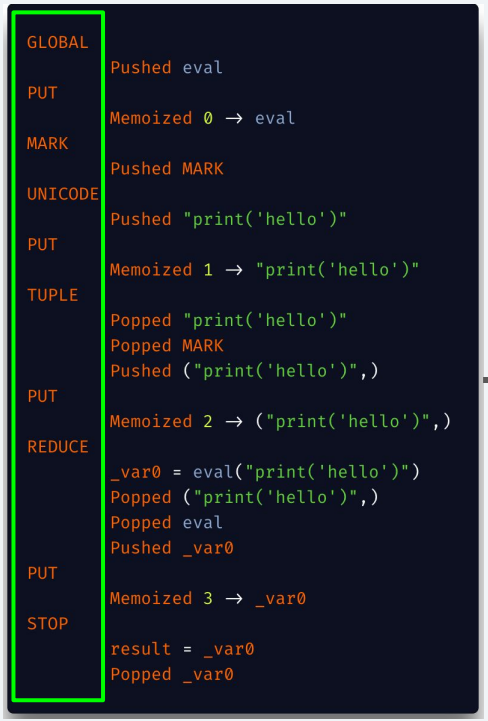
Pickling Example:

```python
import pickle

# Object, some list
my_list = [1, 2, 3]
# Object, some dict
my_dict = {"Evan" : 38, "Jim": 26}
# Object, my custom class
my_class = SomeClass()

# Serialize to pickle files
f = open("example.p", "wb")
pickle.dump(my_list, f)
pickle.dump(my_dict, f)
pickle.dump(my_class, f)
f.close()
```

```python
import torch
from torch import nn

# Define model
class TheModelClass(nn.Module):
    def __init__(self):
        super(TheModelClass, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

# Initialize model
model = TheModelClass()

# Save the model
torch.save(model, "./model")
```

Slicing A Pickle

totally_safe_trust_me.pkl



Disassembled Pickle Opcodes

```
_var0 = eval("print('hello')")
result = _var0
```

Decompiled Python Equivalent

Neighbor! Boom! Big reveal: I'm a pickle. What do you think about that? I turned myself into a pickle! W-what are you just staring at me for, bro. I turned myself into a pickle, Neighbor!

Pickling is not secure

Unpickling (deserialization) has its own instruction set and VM!

Can call into arbitrary Python code

Pickles are a streaming format

“Pickle bombs” have become an attack vector

> **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.

Do you know how pickles are stored? It's *jarring!*

Machine Learning/NLP/CV Model Serialization

PyTorch, TensorFlow, spaCy, arm, scikit learn, Azure ML, theano

Pickling allows for easy serialization of custom models

Models use Python pickling and its variants

Trained models are distributed as pickle files

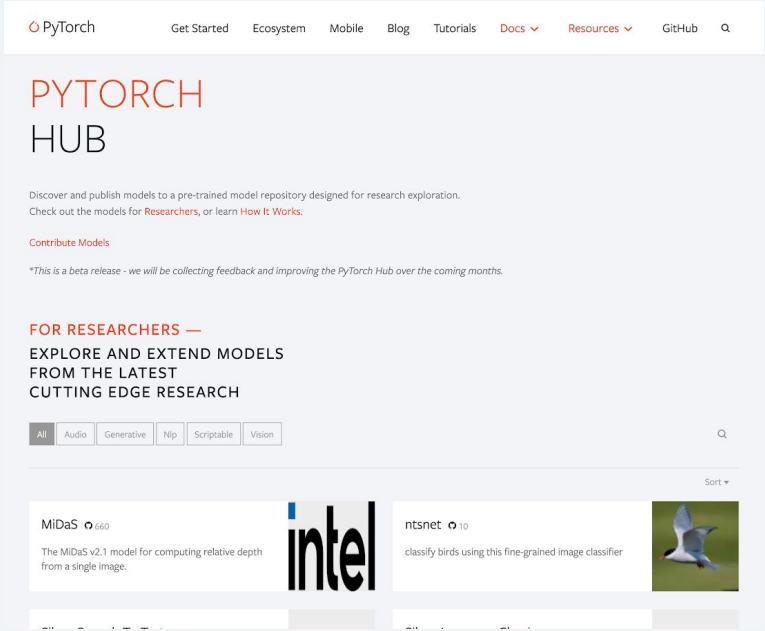
Models are Anonymously Shared Online!

Pickle ACE/RCE well known in the security community

Many ML practitioners have no CS, let alone security, background

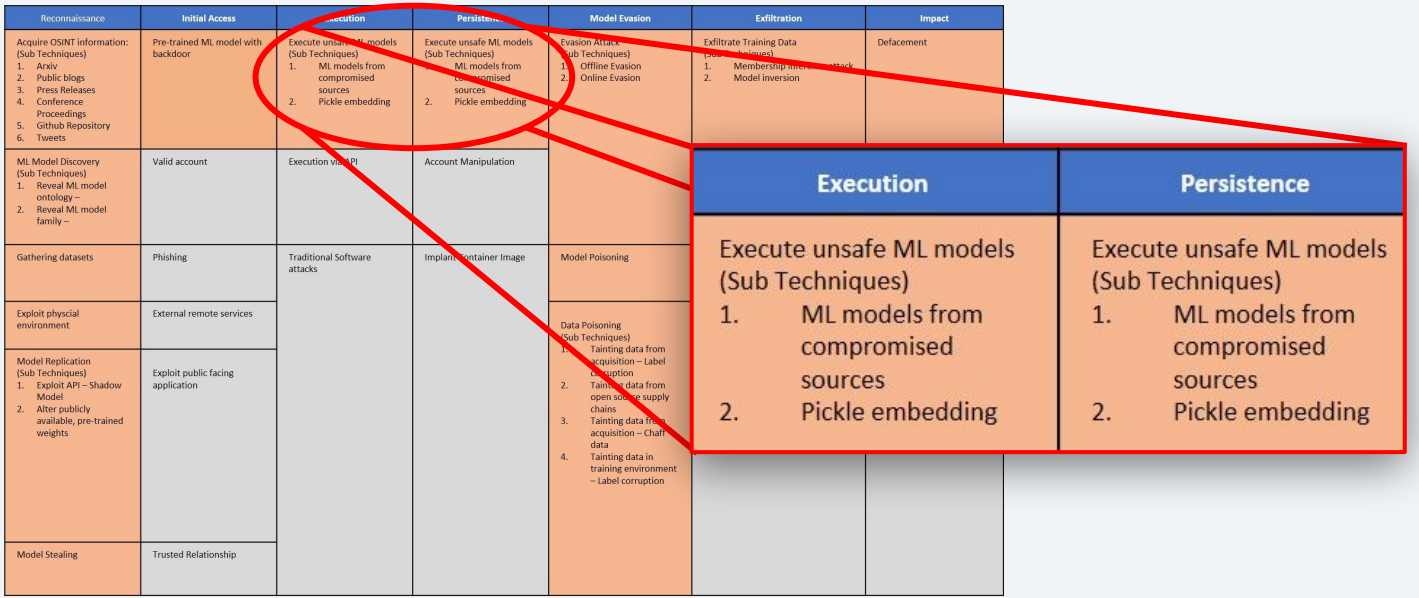
Admonitions to "follow the warnings in the Pickle documentation," but no direct warnings

Pretrained models often shared



In which it is revealed that yet another file format contains a weird machine

Mitre Adversarial Machine Learning Threat Matrix (ATLAS)



Reconnaissance	Initial Access	Execution	Persistence	Model Evasion	Exfiltration	Impact
Acquire OSINT information: (Sub Techniques) 1. Arxiv 2. Public blogs 3. Press Releases 4. Conference Proceedings 5. Github Repository 6. Tweets	Pre-trained ML model with backdoor	Execute unsafe ML models (Sub Techniques) 1. ML models from compromised sources 2. Pickle embedding	Execute unsafe ML models (Sub Techniques) 1. ML models from compromised sources 2. Pickle embedding	Evasion Attack (Sub Techniques) 1. Offline Evasion 2. Online Evasion	Exfiltrate Training Data (Sub Techniques) 1. Membership Inference Attack 2. Model inversion	Defacement
ML Model Discovery (Sub Techniques) 1. Reveal ML model ontology – 2. Reveal ML model family –	Valid account	Execution via API	Account Manipulation			
Gathering datasets	Phishing	Traditional Software attacks	Implant Container Image	Model Poisoning		
Exploit physcial environment	External remote services			Data Poisoning (Sub Techniques) 1. Tainting data from acquisition – Label corruption 2. Tainting data from open source supply chains 3. Tainting data from acquisition – Chaff data 4. Tainting data in training environment – Label corruption		
Model Replication (Sub Techniques) 1. Exploit API – Shadow Model 2. Alter publicly available, pre-trained weights	Exploit public facing application					
Model Stealing	Trusted Relationship					

Execution	Persistence
Execute unsafe ML models (Sub Techniques) 1. ML models from compromised sources 2. Pickle embedding	Execute unsafe ML models (Sub Techniques) 1. ML models from compromised sources 2. Pickle embedding

How do we *dill* with it? Introducing “Fickling”

Divining the meaning of the “F” in “Fickling”: left as an exercise to the reader

Decompiler: Pickle VM → Python AST → Human-Readable Python

Static Analyzer: Detect overtly malicious code
(***e.g.***, **use of** `eval`, `exec`, `os.system`, `subprocess.call`, ***&c.***)

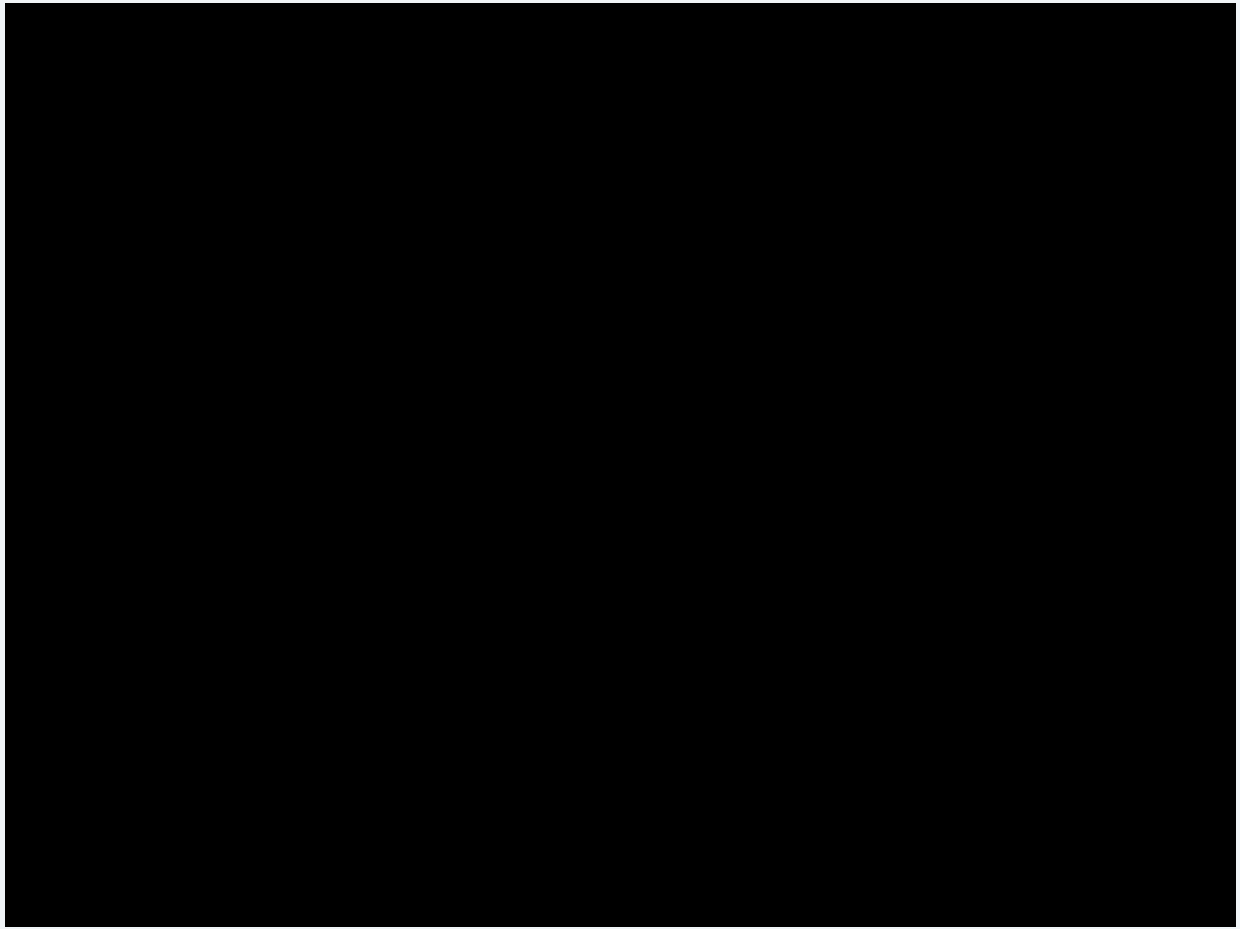
Binary Rewriter:
Inject arbitrary code into an existing pickle (such as an ML model)

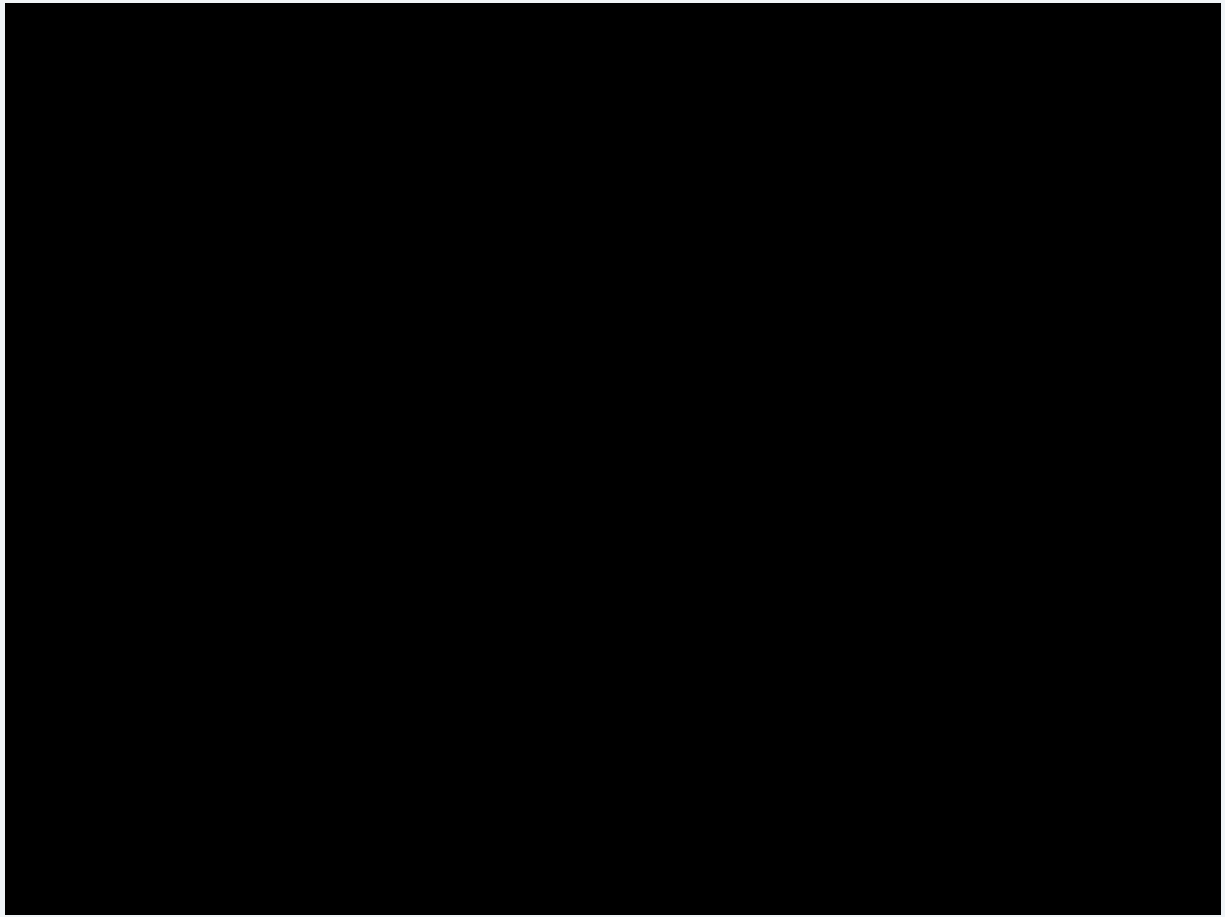
Open-Source: `https://github.com/trailofbits/fickling`

```
pip3 install fickling
```

Proof of Concept Exploits

- **Exfiltrate local files**
  - Proprietary code, proprietary models
- **Potential RCE on proprietary ML systems**
  - Microsoft Azure ML (not yet tested, but plausible)
- **Add arbitrary classification delays (DoS)**
- **Replace trained model with “backdoored” model**
- **Model poisoning attacks**
- **Swap out model parameters or tensors**
  - Time of day, time zone, system locale/language, IP address





We *relish* the thought of a day when pickling will no longer be used to deserialize untrusted files

ReSpOnSiBIE DiScLoSuRe

Reported to PyTorch on January 25th, received reply January 27th

> ***"…Models linked on PyTorch Hub index are vetted for quality and utility but don't do any background checks on the people publishing the model, or carefully audit the code for security before adding a link to the github repository on the PyTorch Hub indexing page. …"***

Basically, a WONTFIX

Caveat sciscitator

What Sci-Kit Learn Has to Say

9.1.1. Security & maintainability limitations

pickle (and joblib by extension), has some issues regarding maintainability and security. Because of this,

- Never unpickle untrusted data as it could lead to malicious code being executed upon loading.
- While models saved using one version of scikit-learn might load in other versions, this is entirely unsupported and inadvisable. It should also be kept in mind that operations performed on such data could give different and unexpected results.

In order to rebuild a similar model with future versions of scikit-learn, additional metadata should be saved along the pickled model:

- The training data, e.g. a reference to an immutable snapshot
- The python source code used to generate the model
- The versions of scikit-learn and its dependencies
- The cross validation score obtained on the training data

This should make it possible to check that the cross-validation score is in the same range as before.

Aside for a few exceptions, pickled models should be portable across architectures assuming the same versions of dependencies and Python are used. If you encounter an estimator that is not portable please open an issue on GitHub. Pickled models are often deployed in production using containers, like Docker, in order to freeze the environment and dependencies.

If you want to know more about these issues and explore other possible serialization methods, please refer to this talk by Alex Gaynor.

Immediate Steps

- **If you must use pickle, use fickling for safer unpickling**
- **Follow best practices**
  - Use the PyTorch state_dict and load_state_dict functions
- **Switch to other file formats**
  - ONNX
  - PPML
  - HDF5
  - SavedModel
  - Protobuf
  - JSON
  - TF Lite
  - CoreML

Most importantly

The “Summoning Demons” Approach

- **We do need to think about novel ML attacks**
  - Countferfit, PrivacyRaven, IBM ART, PyTorchFi
- **But these “demons” need to be “summoned”**
  - Vulnerabilities in file formats, libraries, server architectures, etc.
  - ATLAS Matrix

What Should We Actually Do

- **Make 👋 our 👋 tools 👋 secure 👋 by 👋 default**
- **Minimize access to the model files**
  - Threat modeling is key
- **Sign models and validate signatures**
  - Use model cards and incorporate this
- **Manage your the model's lifecycle**
  - MLFlow
- **Monitor your model. Audit logs**
  - Detect attacks as they happen
- **Validate your inputs**
- **Test your models thoroughly**
  - Unit tests, property tests, fuzzing, etc.

Conclusions

- **Pickling is a convenient but very unsafe serialization**
- **Due to its convenience, pickling has become commonplace in ML**
- **This represents a serious supply chain risk to the ML community**
- **We need to make our tools secure by default**

We *relish* the thought of a day when pickling will no longer be used to deserialize untrusted files

TRAIL OF BITS

Suha S. Hussain
@suhackerr

Carson Harmon
@carsonharmon12

Jim Miller
James.Miller
@trailofbits.com

Evan Sultanik
@ESultanik

🤦 https://github.com/trailofbits/fickling
💬 https://empireslacking.herokuapp.com/
📄 https://www.trailofbits.com/post/never-a-dill-moment-exploiting-machine-learning-pickle-files

TRAIL OF BITS