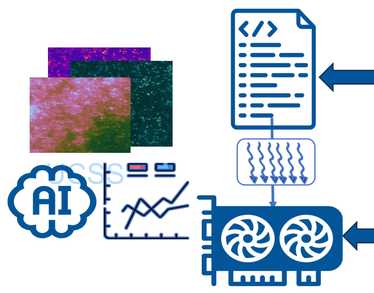
CICI: UCSS: Securing GPU Computing for AI-Driven Scientific Workflows

Yanan Guo (University of Rochester) | Zhenkai Zhang (Clemson University) | Tong Geng (Rice University)

Award Number: 2530649
Keywords: GPU Memory Safety · Vulnerability Detection · Runtime Protection for Scientific Cyberinfrastructure

Project Overview



- **Detecting/repairing vulnerabilities in scientific AI programs.**
- **Developing runtime GPU memory protection techniques.**

- **Detecting/Repairing Vulnerabilities in Scientific AI Programs**
  - Developing a fuzzing framework for AI toolkits on GPU.
  - LLM-assisted repairing method for vulnerabilities in AI toolkits.
- **Developing Runtime GPU Memory Safety Protection Techniques**
  - W^X: code integrity.
  - Secure memory allocator.
  - Enhanced stack canary.
  - Independent GPU ASLR.

Project Goals

- **Open-Source Tools and Detailed Documentation**
  - We will develop and release tools for detecting memory safety vulnerabilities in GPU code.
- **Vulnerability Reports and Database**
  - We will report all the uncovered vulnerabilities to the corresponding development teams as well as maintain a database for the vulnerabilities.
- **Deployment in Scientific CIs**
  - We will collaborate with CI providers to integrate our runtime protection mechanism.
- **Raising Awareness of GPU Security**
  - We aim to greatly raise awareness of GPU security, especially in the scientific CI community.

Ongoing Research I: GPU Fuzzing

- **Constraint-Based Fuzzer + NVIDIA Compute Sanitizer**
  - Uncovered 11 bugs.

Library	Kernel Type	Bug Type	Count
PyTorch	CNN kernels	**Integer overflow**	**7**
TensorFlow	CNN kernels	**Integer overflow**	**2**
PyTorch	LLM / kernels	**Integer overflow**	**2**

```
__global__ void adaptivemaxpool(int oh, int osizeW...) {
  ...
  // potential integer overflow here ↓
  int64_t *ptr_ind = indices + oh*osizeW + ow;
  *ptr_ind = argmax;
}
```

Ongoing Research 2: GPU UBSan (LLVM-Based)

Motivation: The majority of the discovered bugs are **integer overflows**.

The Problem

NVIDIA Compute Sanitizer
- Closed-source; hard to extend.
- Significant performance overhead.
- Unable to detect integer overflows unless the result is used as an index for memory accesses.

Our GPU UBSan

Extended LLVM UBSan to CUDA
- Detects integer overflow—root cause of most bugs.
- Open-source; no hardware changes needed.
- Supports modern AI toolkits, such as PyTorch and TensorFlow.
- Much better performance compared to Compute Sanitizer.

Ongoing Research 3: End-to-End Exploit on LLM Inference

① Attacker prepares a malicious prompt → ② Integer overflow in CUDA → ③ OOB write corrupts KV cache → ④ Victim gets wrong response

Attacker's HTTP request to LLM API

```
{ "prompt": "A".repeat(N),
  "max_tokens": 100 }
```

N is chosen to make **seq_len × batch_size** overflow int32

CUDA Kernel in Transformer

```
int32 idx = seq_len * batch_size;
//idx is negative on overflow
void* ptr = mem_base + idx; ← OOB
```

★ Integer overflow → negative index → write lands in another prompt's KV cache

GPU Memory — Shared KV Cache (batched inference, multiple users)

KV Slot: User A (Attacker) | KV Slot: User B (Victim) | KV Slot: User C | KV Slot: User D | Other Mem Buffer

Corrupted

OOB write

✓ User B's expected response

"The boiling point of water is 100°C at standard atmospheric pressure."

✕ User B's actual (corrupted) response

"3.14159265 KERNEL PANIC NULL NULL NULL ▓▓ [garbled tokens]..."

Publications & Future Work

- **Research sponsored by this award has resulted in publications at NDSS'26, USENIX Security'26, and IEEE S&P'26.**
- **Future work**
  - Runtime GPU memory safety protection mechanisms.
  - Lightweight defenses against data-oriented attacks on GPUs.
  - Memory safety solutions for closed-source GPU code.