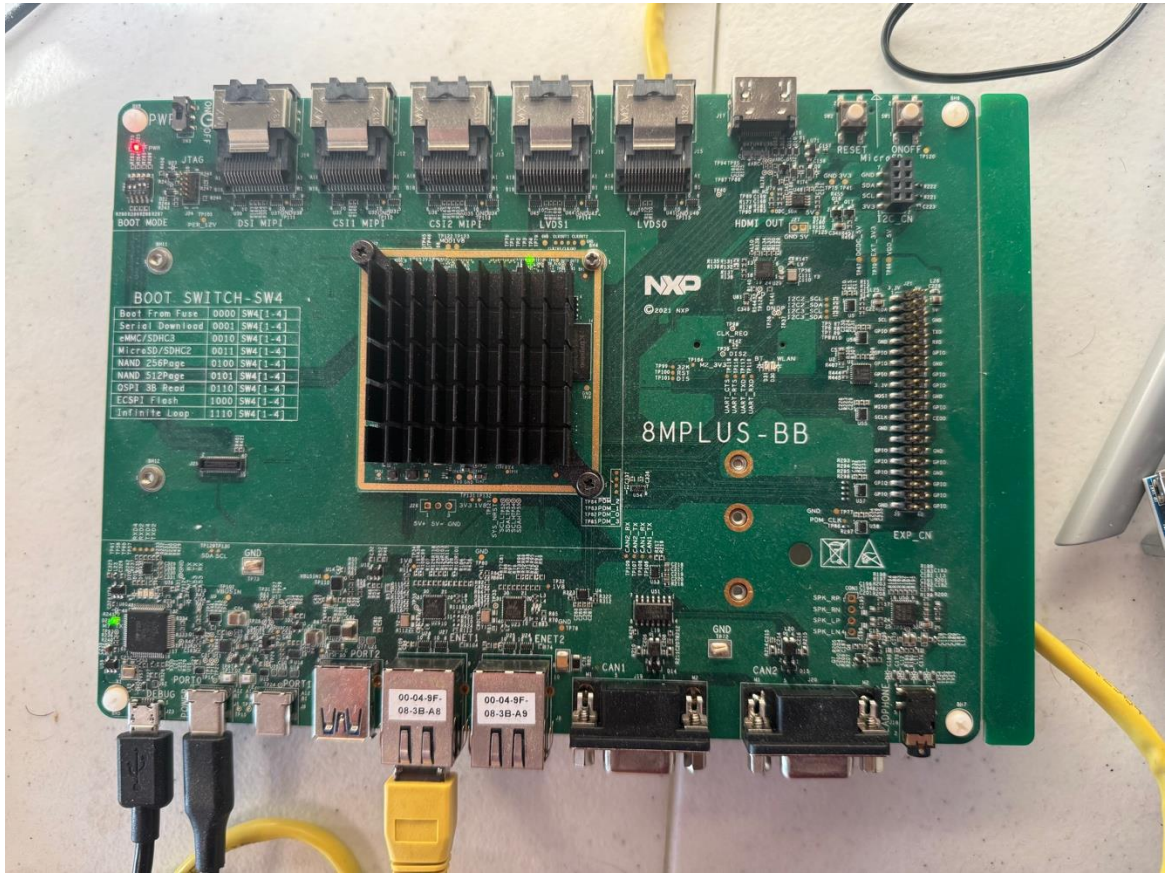
Speed Kills: Exploring Security Aspects of Edge AI Accelerators

Presenter: Srihari Danduri



- What are Edge AI Accelerators?
- Why inference speed is crucial for these devices?
- What are NPU's (Neural Processing Units) ?
- In the Image, below big square heatsink is SOC (System on chip).

What is a confused deputy problem?

In information security, a confused deputy is **a computer program that is tricked by another program (with fewer privileges or less rights) into misusing its authority on the system**. It is a specific type of privilege escalation.

Wikipedia

https://en.wikipedia.org › wiki › Confused_deputy_pro...

Host Memory, Shared between co-processor and Application Processor

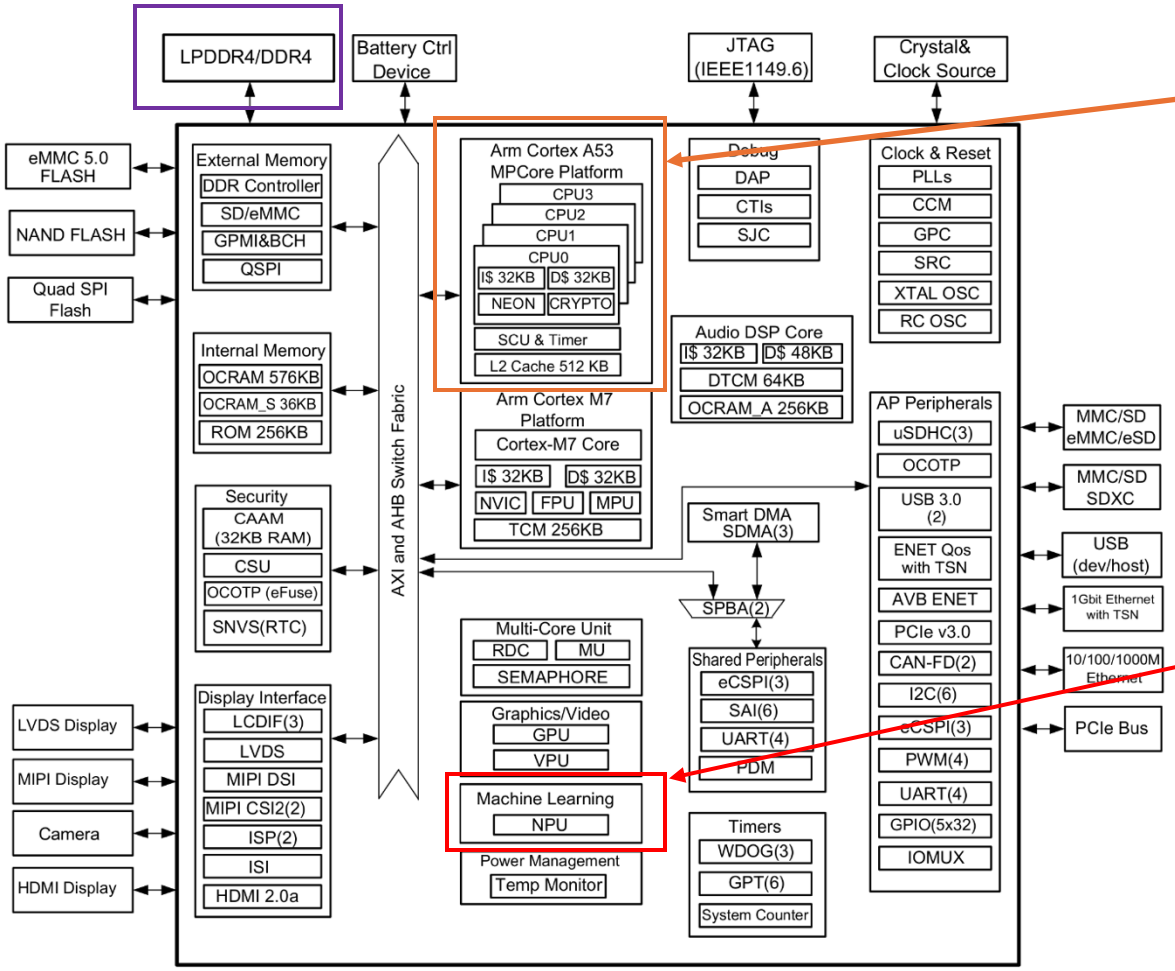


Figure 1-1. Block Diagram

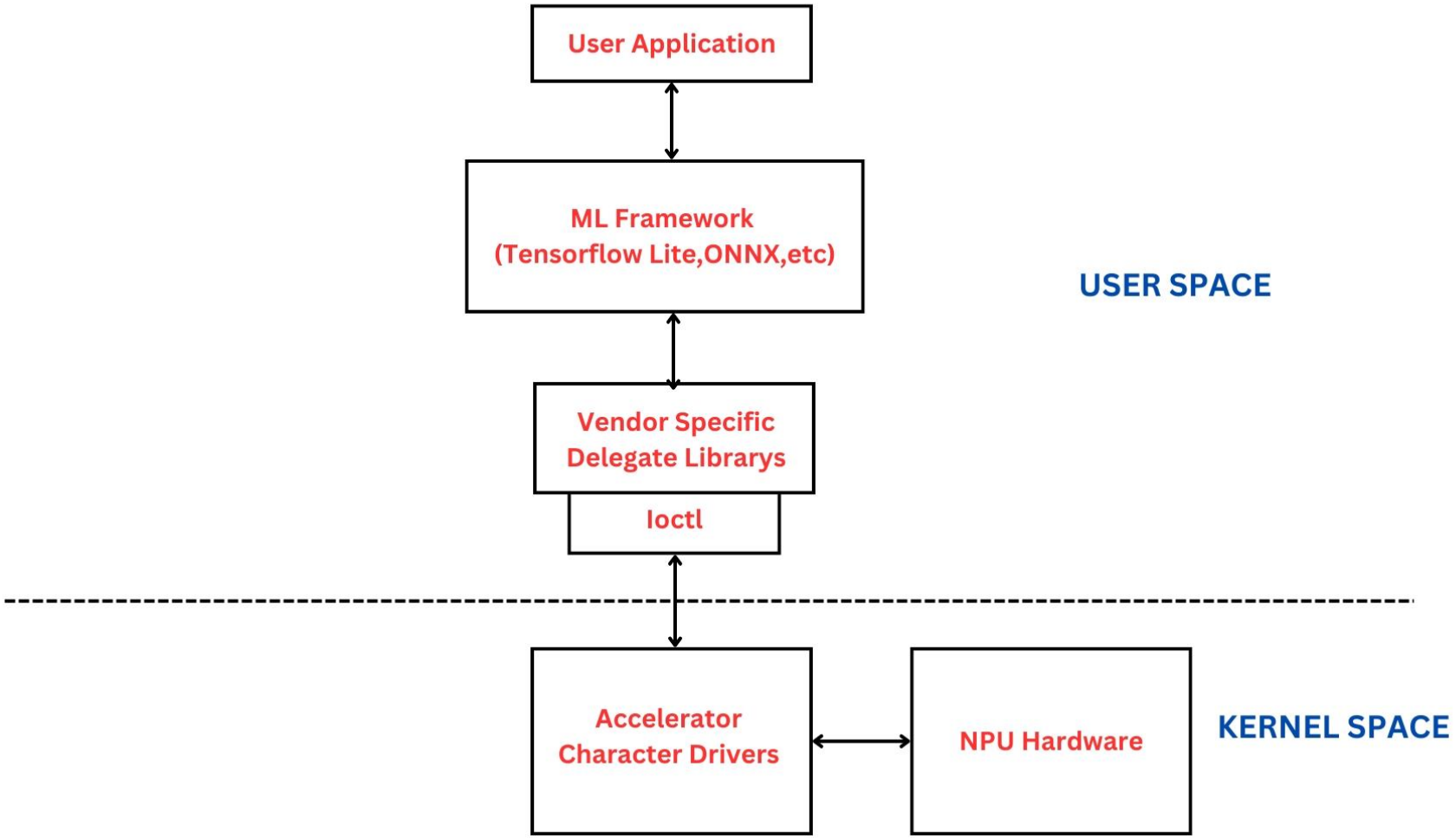
This is where your OS(Linux) and Applications Run

Any Illegal operation done by Application is restricted by OS

Application Machine learning workloads are offloaded to NPU for processing

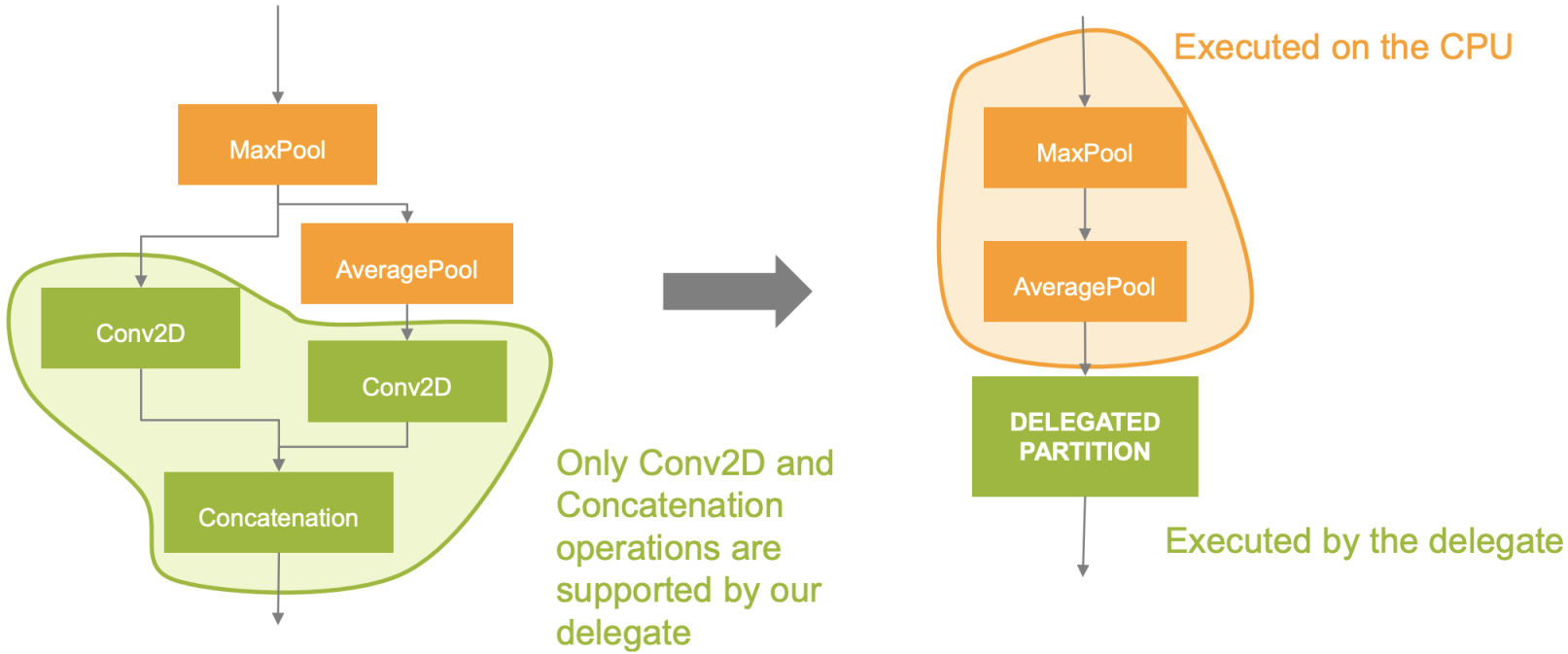
No OS protections when executing Application Machine learning workloads

High-level flow from Application to NPU Hardware



DELEGATE PARTITIONING

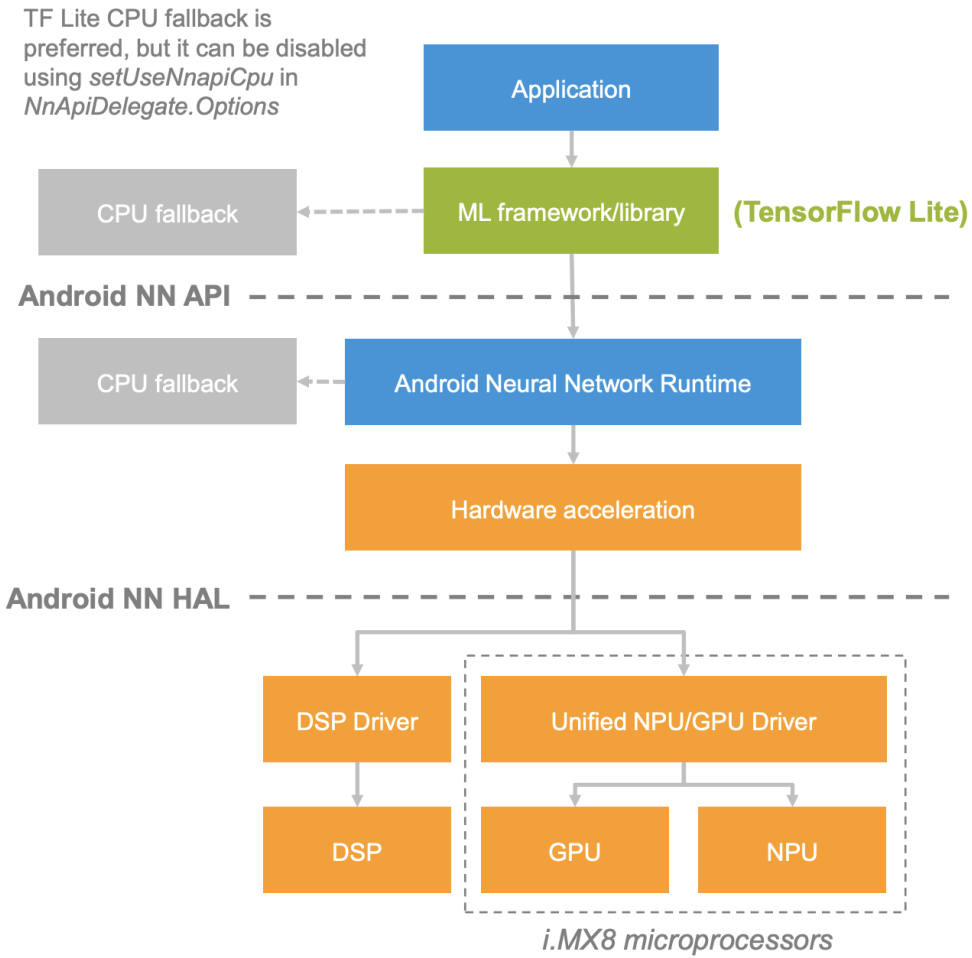
- Graph is partitioned based on op support



Link to official slides from NXP: slides

NN API DELEGATE

- Android C API designed to run machine learning operations on Android devices
- Limited to *float16*, *float32*, *int8* and *uint8*
- Supports acceleration on a GPU, an NPU or a DSP depending on the target device



Link to official slides from NXP: slides

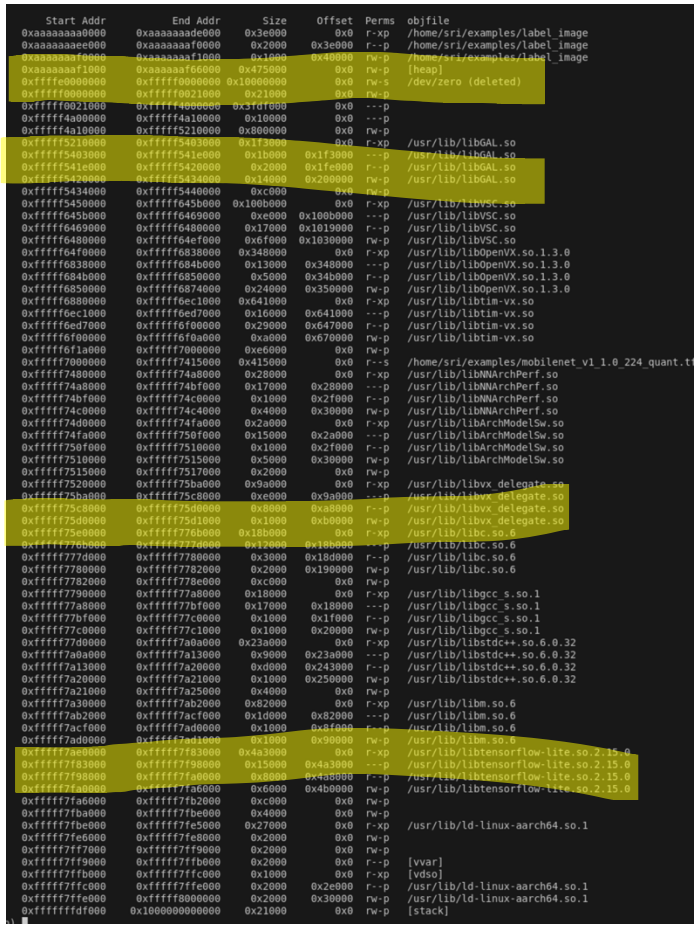
Let's Understand System Design
(Application all the way to NPU Hardware)

```
tflite::ops::builtin::BuiltinOpResolver resolver;
tflite::InterpreterBuilder(*model, resolver)(&interpreter);


auto delegates = delegate_providers.CreateAllDelegates();

for (auto& delegate : delegates) {

const auto delegate_name = delegate.provider->GetName();

if (interpreter->ModifyGraphWithDelegate(std::move(delegate.delegate)) !=

kTfLiteOk) {

LOG(ERROR) << "Failed to apply " << delegate_name << " delegate.";

exit(-1);

} else {

LOG(INFO) << "Applied " << delegate_name << " delegate.";

}

}

for (int i = 0; i < settings->loop_count; i++) {

if (interpreter->Invoke() != kTfLiteOk) {

LOG(ERROR) << "Failed to invoke tflite!";

exit(-1);

}

}
```

Check source code here link

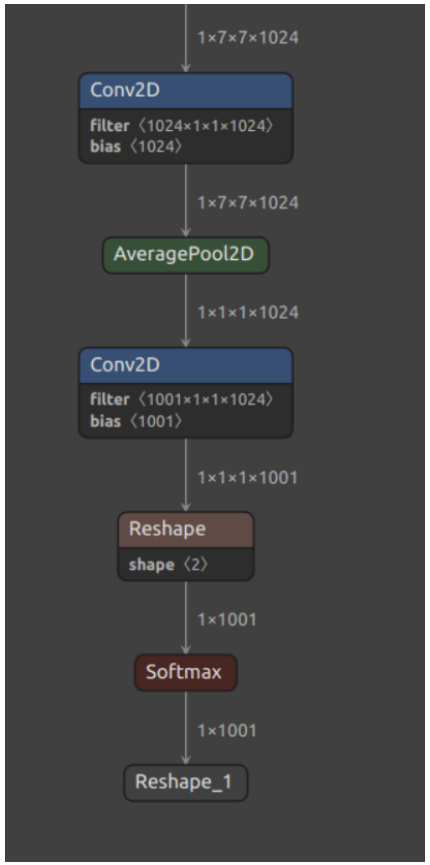
Luanch Application to use NPU

```
export USE_GPU_INFERENCE=0

gdb --nx --args ./label_image -m mobilenet_v1_1.0_224_quant.tflite -i grace_hopper.bmp -l labels.txt --
external_delegate_path=/usr/lib/libvx_delegate.so
```



Let's Understand System Design
(Application all the way to NPU Hardware)

Mobilenet model



Contiguous in User Virtual address space

Input Buffer (1x224x224x3)

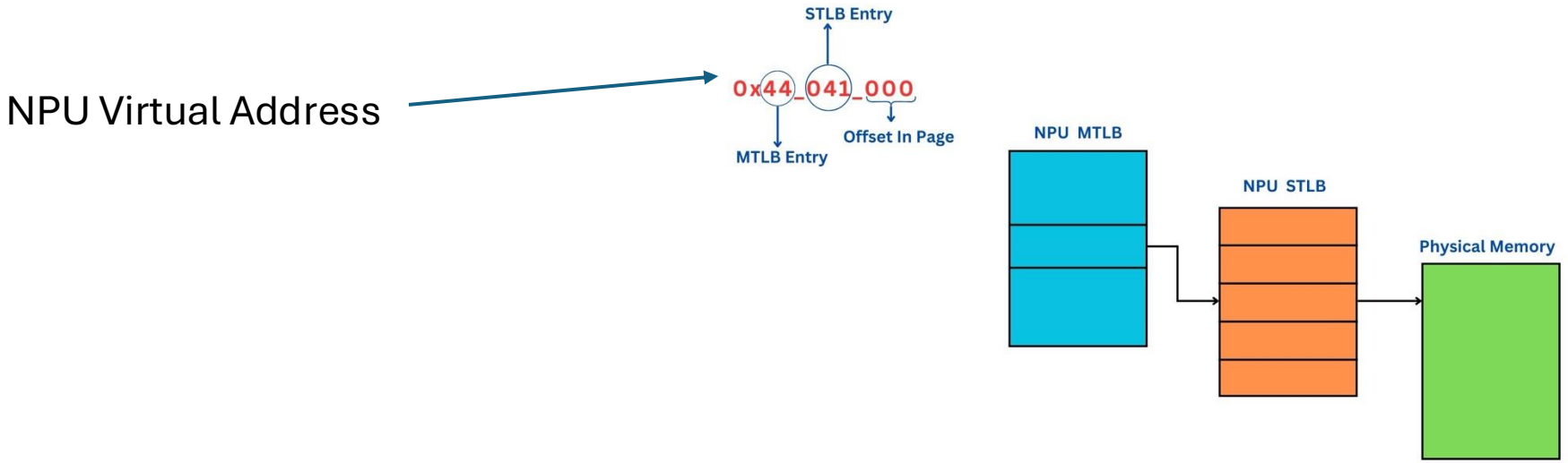
Output buffer 1x1001

Memory Overview

```
[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd034]
[    0.000000] Linux version 6.6.23-gb586a521770e-dirty (sri@sri-nuc) (aarch64-poky-linux-gcc (GCC) 13.2.0, GNU ld (GNU Binutils) 2.42.0.202402
[    0.000000] KASLR disabled due to lack of seed
[    0.000000] Machine model: NXP i.MX8MPlus EVK board
[    0.000000] efi: UEFI not found.
[    0.000000] Reserved memory: created CMA memory pool at 0x00000000c4000000, size 960 MiB
[    0.000000] OF: reserved mem: initialized node linux,cma, compatible id shared-dma-pool
[    0.000000] OF: reserved mem: 0x00000000c4000000..0x00000000ffffffff (983040 KiB) map reusable linux,cma
[    0.000000] OF: reserved mem: 0x0000000000900000..0x000000000096ffff (448 KiB) nomap non-reusable ocram@900000
[    0.000000] OF: reserved mem: 0x0000000056000000..0x0000000057dfffff (30720 KiB) nomap non-reusable optee_core@56000000
[    0.000000] OF: reserved mem: 0x0000000057e00000..0x0000000057ffffff (2048 KiB) nomap non-reusable optee_shm@57e00000
[    0.000000] OF: reserved mem: 0x0000000092400000..0x00000000933fffff (16384 KiB) nomap non-reusable dsp@92400000
[    0.000000] OF: reserved mem: 0x0000000093400000..0x00000000942effff (15296 KiB) nomap non-reusable dsp_reserved_heap@93400000
[    0.000000] OF: reserved mem: 0x00000000942f0000..0x00000000942f7fff (32 KiB) nomap non-reusable vdev0vring0@942f0000
[    0.000000] OF: reserved mem: 0x00000000942f8000..0x00000000942fffff (32 KiB) nomap non-reusable vdev0vring1@942f8000
[    0.000000] Reserved memory: created DMA memory pool at 0x0000000094300000, size 1 MiB
[    0.000000] OF: reserved mem: initialized node vdev0buffer@94300000, compatible id shared-dma-pool
[    0.000000] OF: reserved mem: 0x0000000094300000..0x00000000943fffff (1024 KiB) nomap non-reusable vdev0buffer@94300000
[    0.000000] OF: reserved mem: 0x0000000100000000..0x000000010fffffff (262144 KiB) nomap non-reusable gpu_reserved@100000000
[    0.000000] NUMA: No NUMA configuration found
[    0.000000] NUMA: Faking a node at [mem 0x0000000040000000-0x00000001bfffffff]
[    0.000000] NUMA: NODE_DATA [mem 0x1bf4386c0-0x1bf43afff]
[    0.000000] Zone ranges:
[    0.000000]   DMA      [mem 0x0000000040000000-0x00000000ffffffff]
[    0.000000]   DMA32    empty
[    0.000000]   Normal   [mem 0x0000000100000000-0x00000001bfffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x0000000040000000-0x0000000055ffffff]
[    0.000000]   node   0: [mem 0x0000000058000000-0x000000000923fffff]
[    0.000000]   node   0: [mem 0x0000000092400000-0x00000000943fffff]
[    0.000000]   node   0: [mem 0x0000000094400000-0x00000000ffffffff]
[    0.000000]   node   0: [mem 0x0000000100000000-0x000000010fffffff]
[    0.000000]   node   0: [mem 0x0000000110000000-0x00000001bfffffff]
[    0.000000] Initmem setup node 0 [mem 0x0000000040000000-0x00000001bfffffff]
[    0.000000] On node 0, zone DMA: 8192 pages in unavailable ranges
```

NPU Platform Drivers

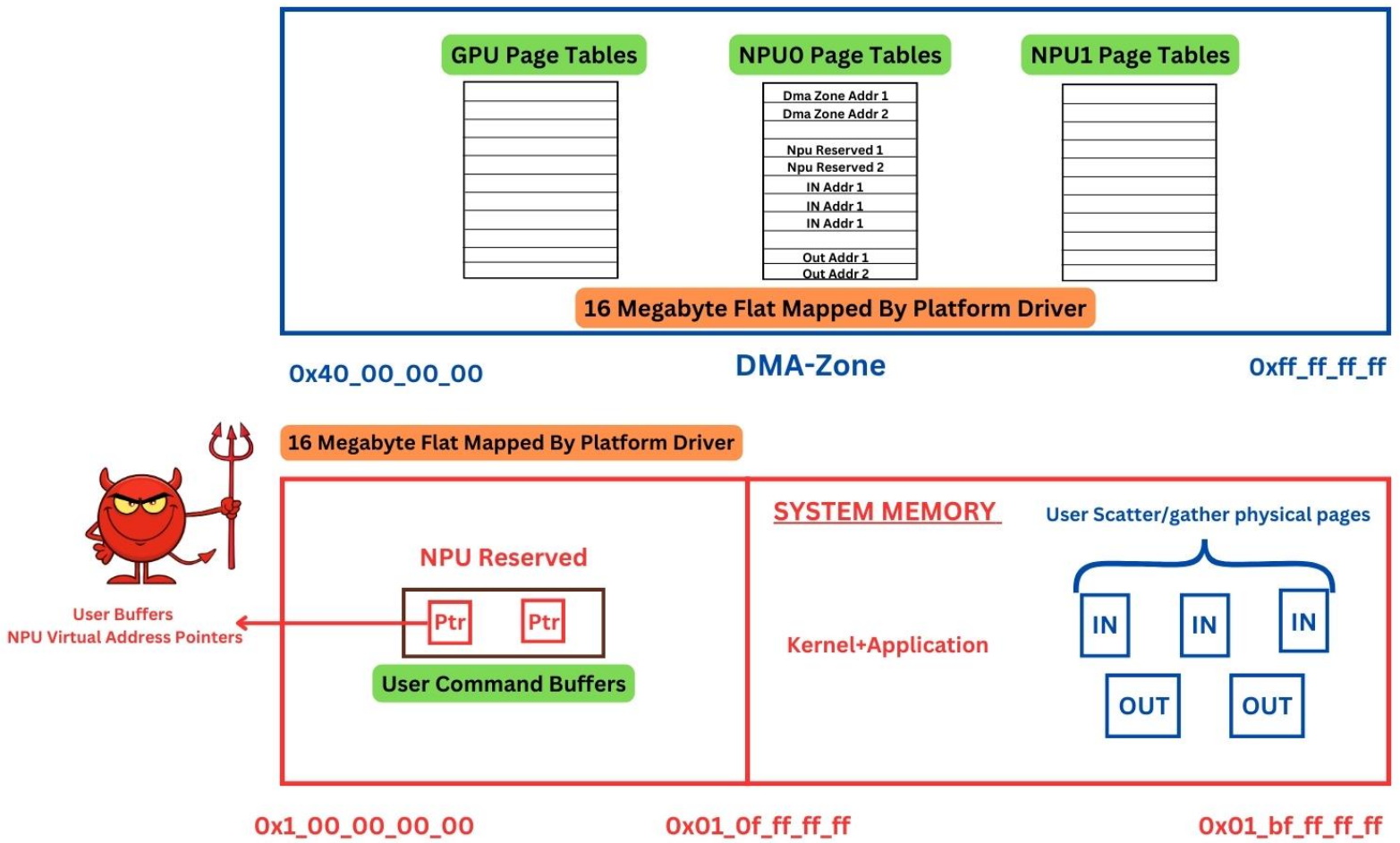
- NPU Platform Drivers probe for supported hardware coming in from the device tree and then allocate and construct page tables.
- These allocated page tables are part of DMA zone.
- The base address of these pagetables are programmed in to MMIO(Memory Mapped regions) of NPU so it know where to do page walk during address translation.



NPU Character Drivers

- Ioctl calls are performed to carveout address space for mapping reserved regions in to user process.
- Application runtime uses this mmapped reserved region to construct command buffers.
- Using Ioctls these command buffers are submitted to NPU to perform computation.
- Ioctl calls are performed to do on-demand page mapping of user buffer pages into NPU pagetables. Also marks those physical pages for DMA capable so DMA controller does the copying.

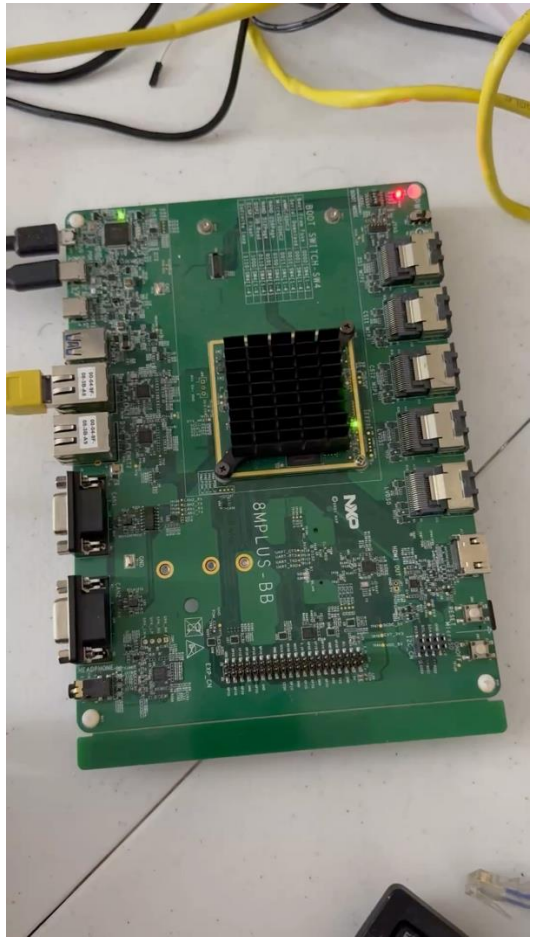
Putting all together



Demo Time

Recorded video

Incase of a live demo disaster



- To run the previous exploit, all you need is 4 bytes of stack corruption.
- Nothing fancy prerequisites to be met to do confused deputy-style attacks using NPU.
- In our case user application can be malicious or it can be a compromised user.

Shortcuts in System design

- To Optimize for Inference speed, vendors used a design with shared memory that can be accessible by the Application, Kernel, and NPU hardware.
- Here input, output buffers and reserved memory act as shared memory for all three.
- Do as much Application ←-→ NPU communication as possible to reduce the overhead of going through Kernel.
- Flat mapping entire regions of memory for speed, this choice only increases the attack surface.

Security Properties Schematic gap between host and NPU:

- On the host each process has its own page tables thereby there is clear separation between processes and the host processor knows what physical memory ranges a process has access to.
- whereas as in NPU there is one set of page tables shared by all processes which kernel driver programs entries, and there is no <process, memory regions it can access information>
- The processing is on a command basis and there is no <processid, command> relationship understood by NPU. Multiple processes can issue commands, and NPU is a shared resource.
- Even if you have protections against kernel memory, there will always be cross-process contamination.

Proposed Solutions

- Kernel driver Sanitization: Have the kernel driver sanitize all the command buffers before committing to the co-processor. They might have to open out the closed source code of command buffer construction and all address offsets in it, and implement sanitization in the kernel driver.
- GPU/NPU Firmware level-fix: Before reading/writing values to and from physical memory have the co-processor send a request to the host processor to confirm the validity of the regions for a given command.

Thank you