Abstract – In this document, we provide a detailed analysis of the "LeftoverLocals" CVE-2023-4969 vulnerability, which has significant implications for the integrity of GPU applications, particularly for large language models (LLMs) and machine learning (ML) models executed on affected GPU platforms, including those from Apple, Qualcomm, AMD, and Imagination.

This document provides valuable insights for cybersecurity professionals, DevOps teams, IT specialists, and stakeholders in various industries. The analysis is designed to enhance the understanding of GPU security challenges and to assist in the development of effective strategies to safeguard sensitive data against similar threats in the future.

I. Introduction

Trail of Bits has disclosed a vulnerability named LeftoverLocals, which allows the recovery of data from GPU local memory that was created by another process. This vulnerability affects Apple, Qualcomm, AMD, and Imagination GPUs and has significant implications for the security of GPU applications, especially large language models (LLMs) and machine learning (ML) models run on the affected platforms.

The vulnerability enables an attacker to listen in on another user's interactive LLM session across process or container boundaries. Moreover, the vulnerability is significant in the context of LLMs and ML models because it can lead to the leakage of sensitive data involved in training these models.

II. Vulnerable environments

The LeftoverLocals vulnerability can be exploited in various environments, including cloud providers, mobile applications, and potentially even in remote attacks.

- **Cloud Providers**: Cloud providers often offer GPU resources to their customers, which are shared among multiple users. In such multi-tenant environments, the LeftoverLocals vulnerability can be exploited if an attacker has access to the same physical GPU as the victim. This could allow the attacker to recover data from the GPU's local memory that was created by another process, leading to significant data leakage. This is particularly concerning for applications that use large language models (LLMs) and machine learning (ML) models, as these applications often handle sensitive data.
- **Mobile Applications**: Mobile devices that use vulnerable GPUs are also at risk. For example, Apple has acknowledged that devices such as the iPhone 12 and M2 MacBook Air are affected by the LeftoverLocals vulnerability.
- **Remote Attacks**: LeftoverLocals vulnerability could potentially be exploited remotely in scenarios where an attacker has compromised a system and gained the ability to run custom code, or in environments where users are allowed to run custom GPU compute applications.

III. LeftoverLocals vs. other vulnerabilities

A. leftoverlocals vs. GPU vulnerabilities

The LeftoverLocals vulnerability is distinct from other GPU vulnerabilities primarily in its method of data leakage through GPU local memory. Unlike many vulnerabilities that exploit specific software bugs or hardware flaws, LeftoverLocals is based on the failure of GPU frameworks to completely isolate memory between processes. This allows an adversary to run a GPU compute application to read data left in the GPU local memory by another user.

Other GPU vulnerabilities might target different aspects of GPU architecture or software, such as buffer overflows, race conditions, or driver-level exploits. These vulnerabilities often require specific conditions to be met or rely on complex interactions between software and hardware.

The leaked data can be substantial enough to reconstruct the models or responses, posing a significant risk to the confidentiality of the processed information.

The severity of the LeftoverLocals vulnerability is high due to several factors:

- **Broad Impact**: The vulnerability affects a wide range of GPUs from major manufacturers like AMD, Apple, Qualcomm, and Imagination Technologies.
- **Data Leakage**: LeftoverLocals can leak significant amounts of data. For instance, on an AMD Radeon RX 7900 XT GPU, it can leak about 5.5 MB of data per GPU invocation, which can amount to about 181 MB for each LLM query. This is sufficient to reconstruct the LLM response with high precision.
- **Ease of Exploitation**: The vulnerability can be exploited by simply running a GPU compute application to read data left in the GPU local memory by another user.

- **Mitigation Challenges**: Mitigating the vulnerability may be difficult for many users. One suggested mitigation is modifying the source code of all GPU kernels that use local memory to store 0 to any local memory locations that were used in the kernel before it ends. However, this might impact performance.
- **Sensitive Data Exposure**: The vulnerability is particularly concerning in the context of AI and machine learning, where sensitive data is often used in training models.

B. *leftoverlocals vs. CPU vulnerabilities*

Spectre and Meltdown are CPU vulnerabilities that exploit "side-channel" attacks, which involve extracting information from the physical implementation of computer systems rather than software bugs or errors.

Spectre tricks other applications into accessing arbitrary locations in their memory. Meltdown, on the other hand, breaks the fundamental isolation between user applications and the operating system, allowing an application to access all system memory, including memory allocated for the kernel.

In terms of severity, all three vulnerabilities are serious as they can lead to unauthorized access to sensitive data. However, they differ in their scope and the nature of the data they can expose. LeftoverLocals primarily affects GPU applications and can lead to the leakage of data from LLMs and ML models. Spectre and Meltdown, on the other hand, can potentially expose any data processed by the CPU, including passwords, encryption keys, and other sensitive information.

The potential consequences of these vulnerabilities are severe:

- They affect almost all CPUs released since 1995, making their impact extremely widespread.
- They can potentially expose any data processed by the CPU, including passwords, encryption keys, and other sensitive information.
- They are hard to detect as the exploitation does not leave any traces in traditional log files.

C. *Similarities*

The vulnerabilities differ in their specific mechanisms and the domains they affect (GPUs for LeftoverLocals and CPUs for Spectre/Meltdown). Spectre and Meltdown are also considered to be more pervasive and difficult to mitigate due to their presence in CPUs used in a vast array of devices over the past couple of decades.

The LeftoverLocals vulnerability shares some similarities with the Spectre and Meltdown vulnerabilities in terms of their implications for security:

- **Data Leakage**: Both LeftoverLocals and Spectre/Meltdown allow unauthorized access to sensitive data. LeftoverLocals enables data recovery from GPU local memory, while Spectre and Meltdown exploit CPU speculative execution to access protected memory.
- **Exploitation of Hardware Features**: Both sets of vulnerabilities exploit hardware features designed for performance optimization—GPU local memory in the case of LeftoverLocals, and speculative execution in CPUs for Spectre and Meltdown.
- **Cross-Process Boundary Violation**: They both violate process isolation guarantees. LeftoverLocals reads data across process or container boundaries on GPUs, and Spectre/Meltdown can read data across application boundaries on CPUs.
- **Affecting Multiple Vendors**: Both vulnerabilities impact products from multiple vendors. LeftoverLocals affects GPUs from Apple, Qualcomm, AMD, and Imagination Technologies, while Spectre and Meltdown affect CPUs from Intel, AMD, and ARM.
- **Complex Mitigation**: Mitigating both vulnerabilities is non-trivial. LeftoverLocals may require changes to GPU kernel code, while Spectre and Meltdown have required a combination of microcode updates, operating system patches, and in some cases, hardware redesigns.
- **Stealthy Nature of Attacks**: Attacks exploiting these vulnerabilities are difficult to detect as they do not leave traditional traces in log files, making it challenging to determine if they have been used in real-world attacks.

IV. LEFTOVERLOCALS EXPLOITATION REQUIREMENTS

- **Shared Access to a GPU**: An attacker needs shared access to a GPU via a programmable interface.
- **Listener-Writer Model**: The exploitation process involves two different programs: a Listener and a Writer. The Writer stores specific values (referred to as "canary values") in local memory, while the Listener reads uninitialized local memory to check for these canary values.
- **Access to Devices**: The attacker needs access to the devices.

A. *Shared Access to a GPU*

The exploitation of the LeftoverLocals vulnerability requires shared access to a GPU, which is a common scenario in multi-tenant environments where multiple users or applications may be utilizing the same physical GPU resources. This can occur in cloud computing platforms, shared data centers, or any system where GPU resources are allocated dynamically to different users or tasks. In such environments, the GPU's local memory is not always properly cleared between different kernel executions or between the usages by different processes. This oversight allows for the possibility that sensitive data from one process could be left in the local memory and subsequently accessed by another process that is scheduled to use the same GPU.

B. *Listener-Writer Model*

The Listener-Writer model is a method used to exploit the LeftoverLocals vulnerability. It involves two different programs: a Listener and a Writer. These programs interact with the GPU's local memory to demonstrate the vulnerability.

The Writer program is designed to intentionally store specific values, referred to as "canary values," in the GPU's local memory. These values are unique and identifiable, serving as markers that can be detected later. The Writer program does not clear these values from the local memory after it finishes its execution.

The Listener program is designed to read uninitialized local memory on the GPU. It scans the local memory looking for the canary values that the Writer program left behind. If the Listener program detects these canary values, it indicates that the local memory was not properly cleared between the execution of different programs.

C. *Access to Devices*

Access to devices is a critical requirement for exploiting the LeftoverLocals vulnerability. Attackers need to have some level of operating system access on the target device to exploit the vulnerability. This access doesn't necessarily need to be root or administrative access; it could be any level of access that allows the attacker to execute their own GPU compute applications.

In the case of Apple devices, the company has acknowledged that devices such as the iPhone 12 and M2 MacBook Air are affected by the LeftoverLocals vulnerability. While Apple has shipped fixes with its latest hardware, millions of existing devices that rely on previous generations of Apple silicon remain potentially vulnerable.

Qualcomm and AMD have also confirmed the impact of the vulnerability on their GPUs and have taken steps to address it. Qualcomm has released firmware patches, and AMD has detailed plans to offer optional mitigations should be released

V. PROCCESS FLOW & POC

The LeftoverLocals vulnerability can be exploited using a method that involves modification, fingerprinting the model, and listening to the LLM output.

A. *Modification*

The first step in exploiting the LeftoverLocals vulnerability involves modifying the GPU kernel code. The researchers at Trail of Bits were able to modify the GPU kernel code to read and write to the GPU's local memory. This allowed them to create a proof-of-concept (PoC) where an attacker can listen into another user's interactive LLM session across process or container boundaries.

B. *Fingerprinting the Model*

Fingerprinting the model involves identifying the specific LLM being used. This can be done by observing the GPU memory usage patterns of the LLM. Different LLMs will have different memory usage patterns, and by observing these patterns, an attacker can determine which LLM is being used. This information can be used to tailor the attack to the specific LLM, increasing the chances of successfully exploiting the vulnerability.

C. *Listening to the LLM Output*

Once the model has been fingerprinted, the attacker can then start listening to the LLM output. This involves repeatedly launching a GPU kernel that reads from uninitialized local memory on the GPU. The attacker scans the local memory looking for specific values that the LLM has left behind. If these values are detected, it indicates that the local memory was not properly cleared between the execution of different programs. This allows the attacker to recover data from the LLM's computations, leading to significant data leakage.

D. *PoC*

The proof-of-concept (PoC) was developed using OpenCL, a framework for writing programs that execute across heterogeneous platforms and built by the researchers at Trail of Bits to demonstrate the LeftoverLocals vulnerability has several key features:

- **Model Fingerprinting**: The PoC involves identifying the specific large language model (LLM) being used by observing the GPU memory usage patterns. Different LLMs have different memory usage patterns, which can be used to determine which LLM is being used.
- **Listening to LLM Output**: The PoC involves repeatedly launching a GPU kernel that reads from uninitialized local memory on the GPU. The attacker scans the local memory looking for specific values that the LLM has left behind. If these values are detected, it indicates that the local memory was not properly cleared between the execution of different programs, allowing the attacker to recover data from the LLM's computations.
- **Data Leakage**: The researchers found that the LeftoverLocals vulnerability can leak approximately 5.5 MB per GPU invocation on an AMD Radeon RX 7900 XT, which, when running a 7B model, adds up to about 181 MB for each LLM query. This is enough information to reconstruct the LLM response with high precision.
- **Cross-Process or Container Boundaries**: The PoC demonstrates that an attacker can listen into another user's interactive LLM session across process or container boundaries. This shows that the vulnerability can be exploited in multi-tenant environments, such as cloud computing platforms, where multiple users share the same physical GPU.
- **Access to Devices**: The PoC requires the attacker to have access to the target device. This could be any level of access that allows the attacker to execute their own GPU compute applications.