A Review of vulnerabilities in GPUs: Related Attacks, Threats, Defenses, and Fixes

Nick Arellano   
*Fisher College of Science and Mathematics*  
*Towson University*Towson, United States of America  
narella1@students.towson.eduNathan Ketterlinus   
*Fisher College of Science and Mathematics*   
*Towson University*   
Towson, United States   
nketter1@students.towson.edu  
Ayomide Owoade   
*Fisher College of Science and Mathematics*  
*Towson University*   
Towson, United States   
aowoade1@students.towson.edu

*Abstract*—As the demand for graphical computing power has skyrocketed over time, the amount of consumer-grade electronics with discrete Graphics Processing Units (GPUs) has risen to match this demand. As GPUs become a more integral part of the average person’s devices, it becomes critical that they do not present themselves as a possible attack vector for malicious actors. This paper will examine various vulnerabilities and backdoors that exist in consumer-grade GPUs, and consider various solutions to these vulnerabilities, weighing the pros and cons in an ever-evolving cybersecurity landscape.

Keywords—GPU, Vulnerabilities, Side-channel Attack, Review, Deep Learning, Encryption

# Introduction

## Introduction to GPU Architecture

As stated above, the broad appeal of a GPU lies within its parallelism. To utilize SIMD processing to its fullest extent, a GPU must be structured in a particular way. While there are several valid, widely used GPU architectures, this paper will quickly go over the Fermi architecture to highlight the basic principles at play. Figure 1 provides a diagram of a sample GPU, as provided by [1].

A diagram of a computer network

Description automatically generated

Fig. 1. Sample GPU architecture, as seen in [1].

GPUs divide computational work into units called threads. 32 threads combine to make a warp. GPUs contain many different cores, called Streaming Multiprocessors (SMs). Fermi architecture-based GPUs typically contain 15 SMs, which each have 32 SIMD execution units, 16 load/store units, and 4 special function units. 2 warp schedulers and 2 instruction dispatch units issue independent instruction from different warps simultaneously. GPUs also commonly have a memory coalescer unit, which allows for more efficient retrieval of information stored in memory. When more than one thread requests the same information from memory, the requests are combined into a single operation, instead of several duplicate operations. This memory coalescing unit, while improving performance drastically, introduces several vulnerabilities that are exploited by the attacks listed below [2] [3].

## What is a Side Channel Attack?

A side channel attack (SCA) is a passive Electrical-level attack [4]. It is a method of extracting sensitive data from a system, such as Advanced Encryption Standard (AES) keys. The data is extracted in a way that leverages the physical attributes of hardware components, such as their power consumption, electromagnetic emanation, timing information, and sound, among others [5], [6]. SCAs typically have two components: the transmitter (typically in a trojan that has snuck into a secure zone), and the receiver (in an unprivileged area) [7]. The receiver is programmed to listen to and interpret physical characteristics of the hardware as a binary output. By inserting a trojan that can manipulate the hardware in a way that the receiver can listen to, a side channel is established, where sensitive info that the transmitter has access to can be smuggled out.

As a quick case study of a side channel attack, we will consider Wang and Zhang’s Correlation-based SCA [8]. Their attack leverages the fact that there is a linear proportionality between kernel execution time and the number of unique cache lines requests generated during a kernel’s execution. Because these cache line requests are dependent on input data and an AES encryption key, the key can be reverse engineered (and thus recovered) by encrypting enough files while analyzing and profiling the GPU. Figure 2 shows the results of this attack for each of the 16 bytes of the encryption key. Notice the singular, massive jump in correlation for each of the 16 bytes, these are the values of the key that were successfully recovered by this method.

A screenshot of a graph

Description automatically generated

Fig. 2. Correlation analysis result of 10000 samples, as seen in [8].

This attack outline provides a simple implementation of an SCA, in hopes to familiarize the reader with what a side channel attack may look like in context.

## What is AES Encryption?

AES is a widely used symmetric encryption algorithm designed to secure sensitive information. It serves a cryptographic standard for protecting data confidentiality and integrity. Symmetric encryption algorithms use a single key for both encryption and decryption. Security is AES encryption’s main aspect, ensuring that even if someone were to try intercept the encrypted data, it would be infeasible to deduce the original content without with the correct key. Side channel attacks exploit information leaked during the computation process, such as power consumption patterns, to infer the sensitive data like encryption key. So, it’s important to emphasize the importance of addressing such vulnerabilities to enhance the overall security of parallel computing hardware systems [9]

AES Encryption consists of a variable number of rounds, depending on key size. For the three most common key sizes: 128, 192 and 256 bits, AES has 10, 12 and 14 rounds, respectively. For AES-128, one block of data is organized as a 4x4 array of bytes, termed the state. Each round is a sequence of four operations: SubByte, ShiftRow, MixColumn, and AddRoundKey, except for the initial and last rounds. The initial round only has an AddRoundKey, and the last round excludes Mixcolumn. All the round keys are derived from a single initial key [5], [10], [11].

# Attacks

## Physical Side Channel vulnerabilities

The first physical SCA we will discuss was proposed by Gonzalez-Gomez et al. [7]. It establishes a side channel through the temperature of the GPU. By putting the GPU under load at strategic times, the transmitter can effectively control the operating temperature of the GPU. Encoding 1 to a high temperature and 0 to a low temperature, the attack was able to achieve a bitstream of 8.75 bits per second with less than a 2% error rate. The main hurdles of an attack of this nature lie in the fact that GPUs typically have thousands of cores, with only one thermal sensor for the entire device. This means that for a detectable temperature rise to occur, thousands of computations must happen simultaneously. In addition, the bit rate of a thermal SCA will always be astronomically low, because time must be provided to allow the GPU to properly cool between each transmission.

Another attack, outlined by Taneja et al., observes that SoCs can leak information through frequency, power, or temperature [12]. Their paper then outlines different methods of recovering a user’s web-based activities, such as visited sites, entered passwords, or search history. The attack abuses stacked SVG filters on certain objects in JavaScript, such as iframes and hyperlinks. Since certain colors take longer to apply than others, a receiver can essentially rebuild screenshots of a user’s activity once the filters are applied, as shown in figure 3.

A table with numbers and symbols

Description automatically generated

Fig. 3. Chrome pixel stealing results (from left to right): Original image, M1 MacBook Air, M2 MacBook Air, Pixel 6 Pro, OnePlus 10 Pro, Nvidia RTX3060, AMD RX 6600, Intel Iris Xe. The table summarizes recovery rates and accuracies, as shown in [12].

Finally, we will consider one last attack that exploits magnetic emanations that the GPU gives off while under load [13]. Maia et al., were able to successfully infer and reconstruct a high-level overview of a neural network topology that the GPU was operating on, as shown in figure 4. These results were achieved with a simple three-dollar induction sensor.

A graph of a graph

Description automatically generated with medium confidence

Fig. 4. Leaked magnetic signal. (left) an induction sensor captures a magnetic signal when a CNN is running on the GPU. Note the strong correlation between the signal and the network steps. Across two steps, the GPU has to synchronize, resulting in a sharp drop of the signal level (highlighted by selected red circles). (right) We can accurately classify the network steps and reconstruct the topology, as indicated by the labels under the x-axis. Here we highlight the signal regions associated with convolutions (conv), batch-norm (BN), Relu non-linear activations (relu), max-pooling (MP), and adding steps together (add), as shown in [13]

While an explanation of a neural network is outside the scope of this paper, one can use common sense to understand that being able to retroactively recover a neural network’s topology for 3 dollars can be of incredible value, and companies that rely on neural networks for income would want to take steps to protect their revenue source.

## Micro-architectural vulnerabilities

The first micro-architectural attack we will look at, proposed by Almusaddar and Naghibijouybari, notes that in systems where the CPU and an integrated GPU, or iGPU are on the same chip (called a system on a chip, or SOC), CPU memory read requests can be predictably stalled with parallel iGPU memory write requests [14]. This attack has the transmitter encode 1s by writing to several different cache lines to quickly fill and drain the write buffer, stalling the CPU. Note that because the requests are to different cache lines, the coalescing effect does not occur. To encode a 0, the transmitter sends the same number of writes, but to the same cache line to avoid filling the write buffer. This creates a detectable side channel which can leak information at a rate of 1.65 kbps @ 0.49% error or 4.41kbps @ 4.32% error.

The second micro-architectural attack is named Trident [15]. Trident was developed because previous SIMT leakage based SCAs largely rely on a positive correlation that simply does not exist on modern hardware. Trident, however, has 3 parts to its attack.

1. Recover the first 4 bytes of an AES key through a negative correlation analysis.
2. Exploit the timing difference between L1 and L2 cache access for a cache collision-based attack.
3. Utilize a chosen plaintext attack to control the number of memory accesses.

It is worth noting that the first part of the attack is independent, while the third part enables the second. This attack allows for the full recovery of an AES key on modern GPUs.

Timing-based SCAs are also prevalent on mobile platforms as well. Karimi et al., outline a sample timing based attack on a Qualcomm Snapdragon, but note that the attack is generalizable to similar mobile platforms [3]. After identifying a correlation between memory load requests and execution time, Karimi et al. were able to launch a test kernel a million times and collect a million timing samples. Using these samples, a correlation analysis allows for full recovery of an AES key. While this attack is very similar to part two of the Trident attack described above, the implications of such an attack on a mobile platform can be dangerous. Moreso than on personal computers and laptops, mobile devices typically contain an abundance of personally identifying information (PII) and personal health information (PHI), which is critical to keep secure.

On the topic of mobile attacks, another attack, as described by Yang et al, outlines a how a trojan application, with only access to GPU performance counters, can effectively become a keylogger by measuring the amount of GPU overdraw that occurs on certain areas of a touchscreen [16]. By mapping these overdrawn areas to a digital keyboard, one can reconstruct a victim’s inputs with over 80% accuracy, as demonstrated in figure 5. This can be thought of similarly to the SVG filters from above, where multiple effects stacked on the same screen location results in a vector of information loss.

A screenshot of a phone

Description automatically generated

Fig. 5. GPU overdraw on popups of key presses in Android. Blue: 1x overdraw; green: 2x overdraw; pink: 3x overdraw; red: 4x overdraw, as seen in [16]

The mitigations for this attack are quite simple and non-transferrable, so they will be covered here briefly. Proposed defenses include disabling popups of key presses, enabling malware detection, and restricting access to GPU performance counter information [16].

In a fifth micro-architecture-based attack, proposed by Liu et al., attackers can infer what victims are doing across virtual machines (VMs) that share the same physical GPU [17]. While these inferences are relatively vague (like being able to differentiate idling from streaming 4k video to deep learning algorithms), This attack relies on the fact that, while separate VMs have their own dedicated virtual GPU, or vGPU, their performance is ultimately bottlenecked by the limitations of the hypervisor’s GPU. By running a simple read-compute-store operation repeatedly and measuring the execution time, an attacker can accurately infer what other types of loads are on the physical GPU, inferring a victim’s activities. This becomes an issue especially when considering the rising prevalence of cloud computing, where separate user’s workloads may be offloaded onto the same remote GPU. On the off chance that a malicious actor is listening in on sensitive or even classified information, even the insight that someone is doing something like deep learning can be critically important.

# Defenses

Thermal channel attacks have a few ways in which they can be prevented. Some proposed defenses include having GPUs randomly solve garbage computations in an effort to add noise to the system, as well as changing the operating frequency of the GPU over regular intervals to make interpreting data much harder [7]. However, these countermeasures introduce significant speed reductions to systems, and must be implemented carefully to maximize GPU performance.

Pixel stealing attacks based on SVG filters are quite easy to defend against. Some proposed countermeasures include:

1. Running the system well under power or thermal budgets so that frequency is never throttled [12].
2. Preventing SVG filters from applying to iframes and hyperlinks [12]. This feature is already implemented in Firefox.

Timing based SCAs on SoCs have also had defenses proposed which would make it much harder to successfully pull off an attack. Among these include prioritizing new memory read requests in the iGPU, and partitioning channels to multiple memory controllers (MCs) to make it harder for the transmitter to send data [14]. In addition, continuously randomizing the width of the coalescing unit can make it harder to execute attacks based on timing differentials [2].

In addition to developing Trident, the developers also proposed TridentShield as a countermeasure to their timing-based correlation attack [15]. They note that randomly coalescing memory requests or introducing bucket-based coalescing techniques can work, but either have limitations or significant computing overhead. In contrast, TridentShield prevents countermeasures by making memory access times constant, instead of variable and exploitable.

Regarding cross VM attacks, Liu et al., recommend incorporating side-channel aware resource scheduling on all GPUs used in a hypervisor, as well as having the hypervisor monitor GPU activities on each vGPU [17].

# References

[1] X. Wang and W. Zhang, “Cracking Randomized Coalescing Techniques with An Efficient Profiling-Based Side-Channel Attack to GPU,” in Proceedings of the 8th International Workshop on Hardware and Architectural Support for Security and Privacy, Phoenix AZ USA: ACM, Jun. 2019, pp. 1–8. doi: 10.1145/3337167.3337169.

[2] E. Karimi, Y. Fei, and D. Kaeli, “Hardware/Software Obfuscation against Timing Side-channel Attack on a GPU,” in 2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), San Jose, CA, USA: IEEE, Dec. 2020, pp. 122–131. doi: 10.1109/HOST45689.2020.9300259.

[3] E. Karimi, Z. H. Jiang, Y. Fei, and D. Kaeli, “A Timing Side-Channel Attack on a Mobile GPU,” in 2018 IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA: IEEE, Oct. 2018, pp. 67–74. doi: 10.1109/ICCD.2018.00020.

[4] D. G. Mahmoud, V. Lenders, and M. Stojilović, “Electrical-Level Attacks on CPUs, FPGAs, and GPUs: Survey and Implications in the Heterogeneous Era,” ACM Comput. Surv., vol. 55, no. 3, pp. 1–40, Mar. 2023, doi: 10.1145/3498337.

[5] C. Luo et al., “Power Analysis Attack of an AES GPU Implementation,” J. Hardw. Syst. Secur., vol. 2, no. 1, pp. 69–82, Mar. 2018, doi: 10.1007/s41635-018-0032-7.

[6] A. A. Ahmed and M. K. Hasan, “Design and Implementation of Side Channel Attack Based on Deep Learning LSTM,” in 2023 IEEE Region 10 Symposium (TENSYMP), Canberra, Australia: IEEE, Sep. 2023, pp. 1–6. doi: 10.1109/TENSYMP55890.2023.10223652.

[7] J. González-Gómez, K. Cordero-Zuñiga, L. Bauer, and J. Henkel, “The First Concept and Real-world Deployment of a GPU-based Thermal Covert Channel: Attack and Countermeasures,” in 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE), Antwerp, Belgium: IEEE, Apr. 2023, pp. 1–6. doi: 10.23919/DATE56975.2023.10137090.

[8] X. Wang and W. Zhang, “pacSCA: A Profiling-Assisted Correlation-based Side-Channel Attack on GPUs,” in 2020 IEEE 38th International Conference on Computer Design (ICCD), Hartford, CT, USA: IEEE, Oct. 2020, pp. 525–528. doi: 10.1109/ICCD50377.2020.00094.

[9] Y. Gao, Y. Zhou, and W. Cheng, “Efficient electro-magnetic analysis of a GPU bitsliced AES implementation,” Cybersecurity, vol. 3, no. 1, p. 3, Feb. 2020, doi: 10.1186/s42400-020-0045-8.

[10] M. Sabbagh, Y. Fei, and D. Kaeli, “A Novel GPU Overdrive Fault Attack,” in 2020 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA: IEEE, Jul. 2020, pp. 1–6. doi: 10.1109/DAC18072.2020.9218690.

[11] C.-S. Chen, S.-Y. Chiu, and S.-Y. Li, “AES Encryption Method Based on Chaotic Cryptography,” in 2023 International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan), PingTung, Taiwan: IEEE, Jul. 2023, pp. 293–294. doi: 10.1109/ICCE-Taiwan58799.2023.10226728.

[12] H. Taneja, J. Kim, J. J. Xu, S. van Schaik, D. Genkin, and Y. Yarom, “Hot Pixels: Frequency, Power, and Temperature Attacks on GPUs and ARM SoCs.” arXiv, May 22, 2023. Accessed: Nov. 06, 2023. [Online]. Available: http://arxiv.org/abs/2305.12784

[13] H. T. Maia, C. Xiao, D. Li, E. Grinspun, and C. Zheng, “Can one hear the shape of a neural network?: Snooping the GPU via Magnetic Side Channel.” arXiv, Sep. 15, 2021. Accessed: Nov. 06, 2023. [Online]. Available: http://arxiv.org/abs/2109.07395

[14] G. Almusaddar and H. Naghibijouybari, “Exploiting Parallel Memory Write Requests for Covert Channel Attacks in Integrated CPU-GPU Systems.” arXiv, Jul. 30, 2023. Accessed: Nov. 06, 2023. [Online]. Available: http://arxiv.org/abs/2307.16123

[15] J. Ahn et al., “Trident: A Hybrid Correlation-Collision GPU Cache Timing Attack for AES Key Recovery,” in 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Korea (South): IEEE, Feb. 2021, pp. 332–344. doi: 10.1109/HPCA51647.2021.00036.

[16] B. Yang, R. Chen, K. Huang, J. Yang, and W. Gao, “Eavesdropping user credentials via GPU side channels on smartphones,” in Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne Switzerland: ACM, Feb. 2022, pp. 285–299. doi: 10.1145/3503222.3507757.

[17] S. Liu, Y. Wei, J. Chi, F. H. Shezan, and Y. Tian, “Side Channel Attacks in Computation Offloading Systems with GPU Virtualization,” in 2019 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA: IEEE, May 2019, pp. 156–161. doi: 10.1109/SPW.2019.00037.