CS 758: Advanced Topics in Computer Architecture

Lecture #9: GPU & Accelerator Security

Professor Matthew D. Sinclair

Announcements

- HW1
  - Lots of issues following directions about format of submission
  - Next HW:
    - New format (tree structure) for what needs to be submitted
    - Created script for you to verify your submission format is correct
  - Hopefully will release HW1 grades tonight
- HW2
  - Builds on HW1
  - Due date + options will be posted on Piazza
  - Need CHTC access – all **please verify you can log into learn.chtc.wisc.edu**
  - Before Thursdays lecture (guest lecture on CHTC)
  - Bring laptop on Thursday – will be interactive!

Announcements (Cont.)

- Midterms
  - Grading electronically on Gradescope
  - Make sure you sign up for an account
  - Will return to these after releasing HW1 grades
- Preliminary Project Ideas
  - I will be replying to these throughout the week
  - Feel free to schedule a meeting or come to office hours if you want to discuss

Challenges for Designing GPU Security Attacks

- GPUs have massive parallelism

  Programmers control how many TBs run – run 1 attacker and 1 victim TB/SM

  Result: can't rely on other threads to obfuscate

- GPUs have limited amount of cache per thread

  Since running a reduced number of threads/TBs, this is less of an issue

  Use microbenchmarks to identify associativity, size, etc. – write targeted tests

- Additional levels of indirection (e.g., warp schedulers)
- Undocumented hardware features

  Reverse engineer GPU hardware features

At this point you can use very similar attacks to CPUs

GPU Attacks

1. Establish co-location
   - Use 2 GPU streams (1 for attacker, 1 for victim)
   - Size number of TBs per stream to be equal to # of SMs
     - Ensure that attacker and victim are co-located on each SM

2. Establish covert channel
   - Offline profiling: determine L1 (or L2) size, associativity, etc.
   - Victim and attacker allocate arrays with same size as L1 cache (or set) + access
   - If both accessing, should miss – time latency of access

GPU Attacks

3. Can run similar attacks for functional units
   - Access latency for certain functional units (e.g., sinf) relatively consistent
   - Again, time with and without contention – tells attacker info about victims use of functional unit
   - Breakdown of warp schedulers and functional units varies per GPU arch
     - Need to make some slightly modifications to account for this

Additional Topics

- Border Control [MICRO ‘15]
  - Different companies design different accelerators
  - Issue: (potentially malicious) accelerator may access memory it shouldn’t
  - Solution: only allow accelerators to access pages they should be allowed to
- Rendered Insecure [CCS ‘18] / Unveiling Keystrokes [NDSS ‘19]
  - Similar to today’s paper, spy monitors GPU side channel
  - Issue: true graphics applications communicate 1 frame at a time
  - Can use this to infer things like passwords or other text

Additional Topics (Cont.)

- Grand Pwning Unit [S&P ‘15]
  - Can extract (Qualcomm) GPU side channel info leaked by browser extensions
- GPUGuard [ICS ‘19]
  - Use machine learning to identify and close GPU side channels
    - Relies on attacker patterns matching something they trained on
  - 8-23% overhead – ok?
  - 9% false positives – ok?

Additional Topics (Cont.)

- CUDA Leaks [TECS ‘15]
  - GPU didn’t zero out memory values until process exited
  - Read another process’s shared memory values for overlapped processes
  - Also able to break AES encryption by accessing global memory
- Confidentiality Issues in Virtualized Environment [FC ‘14]
  - GPUs didn’t always clear memory between kernels or processes
  - Result: attackers could simply read GPU global memory, extract secrets!

Fixed in newer GPUs with better virtualization support

Conclusion

- GPU security is an on-going, rapidly evolving field
  - Very little research before 2017
- Idea:
  - Often take ideas from CPU security and attacks, apply to GPUs …
    - Need to account for differences (e.g., massive GPU parallelism)
  - … Or exploit on poor process isolation in GPUs
    - GPUs recently adding more multiprogramming support – easier to attack now
  - Newer GPUs include more and better support for security
- Lots of opportunities for interesting research!