Poster: Side-channel Inference of User Activities in AR/VR Using GPU Profiling

Seonghun Son[*] Chandrika Mukherjee[◇] Reham Mohamed Aburas[†] Berk Gulmezoglu[*] Z. Berkay Celik[◇]
[*]Iowa State University [◇]Purdue University [†]American University of Sharjah
Emails:{seonghun, bgulmez}@iastate.edu, {cmukherj, zcelik}@purdue.edu, raburas@aus.edu

Abstract

Over the past decade, AR/VR devices have drastically changed how we interact with the digital world. Users often share sensitive information, such as their location, browsing history, and even financial data, within third-party apps installed on these devices, assuming a secure environment protected from malicious actors. Recent research has revealed that malicious apps can exploit such capabilities and monitor benign apps to track user activities, leveraging fine-grained profiling tools, such as performance counter APIs. However, app-to-app monitoring is not feasible on all AR/VR devices (e.g., Meta Quest), as a concurrent standalone app execution is disabled. In this poster, we present OVRWATCHER, a novel side-channel primitive for AR/VR devices that infers user activities by monitoring low-resolution (1Hz) GPU usage via a background script, unlike prior work that relies on high-resolution profiling. OVRWATCHER captures correlations between GPU metrics and 3D object interactions under varying speeds, distances, and rendering scenarios, without requiring concurrent app execution, access to application data, or additional SDK installations. We demonstrate the efficacy of OVRWATCHER in fingerprinting both standalone AR/VR and WebXR applications. OVRWATCHER also distinguishes virtual objects, such as products in immersive shopping apps selected by real users and the number of participants in virtual meetings, thereby revealing users' product preferences and potentially exposing confidential information from those meetings. OVRWATCHER achieves over 99% accuracy in app fingerprinting and over 98% accuracy in object-level inference.

I. CONTENT

This work [1] was accepted to the **2026 Network and Distributed System Security Symposium (NDSS)** and is scheduled for publication in 2026. The full bibliographic information, including the title, authors, venue, and publication year, is provided above. The paper has been assigned the following DOI: https://dx.doi.org/10.14722/ndss.2026.231302.

As the final publisher version is not yet available, we additionally provide a link to the corresponding arXiv preprint: https://arxiv.org/pdf/2509.10703. The abstract included in this submission is identical to the abstract of the accepted paper.

REFERENCES

[1] Seonghun Son, Chandrika Mukherjee, Reham Mohamed, Berk Gulmezoglu, and Z Berkay Celik. Side-channel inference of user activities in ar/vr using gpu profiling. In *The Network and Distributed System Security Symposium (NDSS)*, 2026.

Side-channel Inference of User Activities in AR/VR using GPU Profiling

NDSS SYMPOSIUM/2026

Seonghun Son[1], Chandrika Mukherjee[2], Reham Mohamed Aburas[3], Berk Gulmezoglu[1], Z. Berkay Celik[2]

Iowa State University[1] Purdue University[2] AUS American University of Sharjah[3]

Introduction

❖ **Motivation:**

- **Problem**
  - AR/VR apps handle private user activity information (e.g., shopping preferences, meetings, browsing, etc.)
  - Prior XR side-channels often require a concurrent app and/or a high-resolution profiling tool
- **Key insight**
  - On Meta Quest, **low-resolution GPU profiling (1Hz)** can leak user activity
- **Why does it matter?**
  - Enables stealthy inference of :
    - Which standalone/WebXR app site is being used
    - Type of 3D object a user is interacting with
    - Private context, like meeting participant count

❖ **Background**

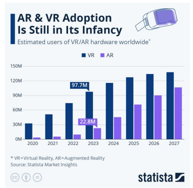
- **Usage of AR/VR devices**

Medical

Education

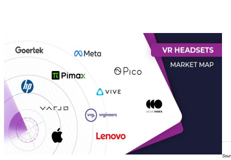
Industry

Entertainment

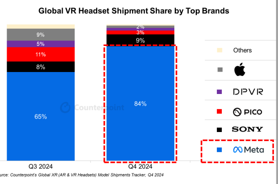


Estimated User Growth

- **Market share**



AR/VR Market Map



Market Share of AR/VR devices

- **Hardware Features**

Meta Quest with Qualcomm Snapdragon XR2 Gen2 chip

OVR Metrics Tool: GPU Profiler Provided by 3rd Third-party App

- **Related work**

Side-Channel Type	Side-channel primitive	Extracted Attributes (# Labels)	Resolution (Hz)	Standalone	AR/VR
Physical	Facial vibration monitoring belt	Gender (2), User(27) Body fat (-)	203	x	VR
	Power monitoring device	App (10), Website(10), Audio (5)	100	✓	VR
Montion/ Gesture Sensors	Controller	Keystrokes (38)	60	✓	VR
	IMU	Keystrokes (60)	72	x	VR
	Camera	Keystrokes (4)	30	✓	VR
	IMU (Accelerometer/Gyro)	Digits (10)	1000	x	VR
System-level APIs	Performance Counter	Gesture (5), Voice (5), Digits (10), App (12) †	60	x	AR/VR
OVRWatcher	**GPU profiler (in-built)**	**VR Object (35), App (100), Website (100), Avatar (10)**	**1**	✓	**AR/VR**

Comparison of related AR/VR side-channel attacks and OVRWATCHER.

Threat Model

❖ **Attacker model**

- User installs a seemingly benign app
- App launches built-in GPU profiler and keeps it running in the background
- Collected GPU traces are stored and transmitted for ML inference

❖ **Constraints addressed**

- No concurrent app execution needed
- Selecting only the most informative GPU metrics that capture real-time user interactions
- Works with 1Hz sampling rate (harder to detect & block)

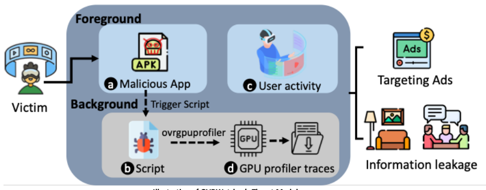
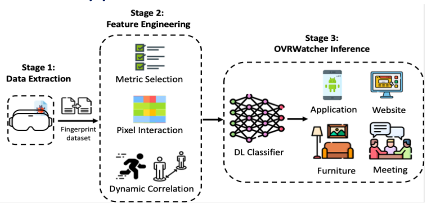


Illustration of OVRWatcher's Threat Model

Methodology

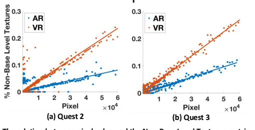
❖ **OVRWatcher pipeline**



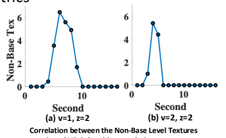
- **Data extraction**
  - Run built-in *ovrgpuprofiler* in the background
  - Collect sequences of GPU counters at 1Hz
- **Feature/metric selection**
  - Select informative subset (avoid missing values)

Category	Metric
GPU Utilization	GPU Frequency, GPU Bus Busy, Preemptions / second, Avg Preemption Delay
Stalls	Vertex Fetch Stall, Texture Fetch Stall, Texture L2 Miss, Stalled on System Memory
Memory Access	Vertex Memory Read (Bytes/Second), SP Memory Read (Bytes/Second), Global Memory Load Instructions, Global Buffer Data Read Request BW (Bytes/sec), Global Buffer Data Read BW (Bytes/sec), Global Image Uncompressed Data Read BW (Bytes/sec), Bytes Data Write Requested, Bytes Data Actually Written
Shader/ Instruction	Vertex Instructions / Second, Local Memory Store Instructions, Avg Load-Store Instructions Per Cycle, Avg Bytes / Fragment, L1 Texture ache Miss Per Pixel
Geometry/ Rasterization	Pre-clipped Polygons/Second, Prims Trivially Rejected, Prims Clipped, Average Vertices/Polygon, Average Polygon Area
Texture/Filtering	Nearest Filtered, Anisotropic Filtered, Non-Base Level Textures

  - Pixel and Speed with GPU metrics



The relation between pixel value and the Non-Base Level Textures metric.



Correlation between the Non-Base Level Textures metric and VR Cube object rendering.

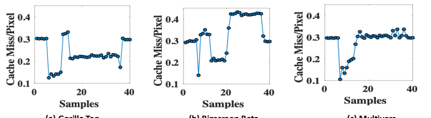
- **Inference models**
  - CNN/ LSTM on sequences
  - RF/ SVM on summary stats (mean, std, min, max)

Takeaway: GPU counters strongly track rendering/pixel-level behavior. For example, Non-Base Level Texture correlates with object size/speed.

Case Studies

❖ **Case Study 1: Standalone app fingerprinting**

- 100 popular Quest apps in the Meta Store

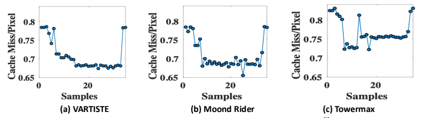


- Achieve **99.3% (CNN)**, **98.6% (LSTM)**, **99.5% (RF)** accuracy

Takeaway: More than **99%** Meta Quest app detection without concurrent app running in the foreground and with a low resolution

❖ **Case Study 2: WebXR app fingerprinting**

- 100 WebXR experiences in the browser

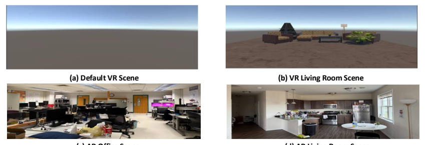


- Achieve **97.6% (CNN)**, **96.4% (LSTM)**, **99.0% (RF)** accuracy

Takeaway: **99%** WebXR detection even with the 2D plane in the browser, utilizing a 1Hz low sampling rate.

❖ **Case Study 3: Virtual Object Detection**

- Identify which product/furniture prefab is rendered (35)

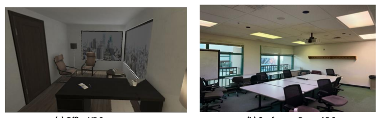


- Achieve **96.5% (CNN)**, **92.6% (LSTM)**, **98.1% (RF)**

Takeaway: Low-resolution GPU metrics can reveal which virtual object is rendered across VR and AR scenes, not just the app.

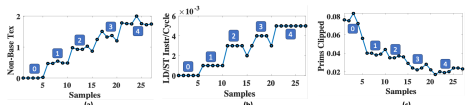
❖ **Case Study 4: Meeting participant inference**

- Infer the number of participants in a meeting-like scene



Experiment scenes for meeting room inference. (a) Office VR scene and (b) conference room scene

- Step-like behavior in GPU metrics as avatars increase



Example of a fingerprint dataset of meeting participant inferences in the default VR scene setup.

- Achieve **100.0% (CNN)**, **90.0% (LSTM)**, **100% (RF)**

Takeaway: GPU metrics exhibit step-like patterns as the number of avatars increases. This enables the inference of participants.

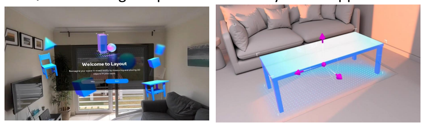
Open World User Interaction

❖ **Goal**

- Validate that ***OVRWatcher*** still works when real users freely interact with AR content in an open-world setting

❖ **Setup**

- Meta Quest 3S using the pre-built Meta Layout AR app



- Participants: 6 participants with IRB approval
- Objects: 5 furniture items, chair, sofa, table, desk, and bed

❖ **Scenario**

- Static : render for 5s, remove for 5s, repeated 10 times per object per participants
- Dynamic: user selects and drags an object into view for 10s, then drags out, repeated 5 times per object per participant

❖ **Results**

Scenario	Model	Test (%)	5-Fold (%)
Static Interaction	RF	83	86 ± 6
	SGB	82	88 ±6
	SVN	30	38 ±4
Dynamic Interaction	RF	81	77 ±5
	XGB	67	66 ±9
	SVM	78	80 ±10

Takeaway: Even with natural head and hand motion, low-resolution GPU metrics can still reveal which virtual object a user is viewing and manipulating.

Countermeasures and Limitations

❖ **Countermeasures**

- **Restrict GPU Profiler Access**
  - GPU profiler APIs are behind developer mode and app store approval, so normal apps cannot sample metrics
- **Dynamic Detection**
  - Detect repetitive profiler polling patterns at runtime and warn the user or throttle access with minimal overhead
- **Noise Injection**
  - Add a controlled dummy GPU workload or random rendering to blur fingerprints

❖ **Limitations**

- **Applicability to Diverse Devices**
  - Different XR platforms expose different profiler interfaces and metrics
- **Generalizability**
  - Multi-app usage and dynamic scenes can reduce accuracy

Conclusion

❖ **Conclusion**

- Multi-app Built-in GPU profilers create a measurable privacy leakage surface in XR
- Distinct GPU fingerprints enable inference of **standalone apps**, **WebXR scenes**, **objects**, and **meeting participation size**
- Attacks can remain effective even at a low **1 Hz sampling rate**
- Practical defenses need platform changes: restrict profiler access, detect abnormal polling, and consider noise injection with performance tradeoffs

❖ **Responsible Disclosure**

- We responsibly disclosed our findings to Meta through the Meta Bug Bounty program, and Meta's security team awarded us a Meta Bounty Award.

Paper Here!