BlackMirror: Preventing Wallhacks in 3D Online FPS Games

Seonghyun Park, Adil Ahmad* and Byoungyoung Lee
Seoul National University and Purdue University*
FPS Games
Suffer from cheaters
Alice and Eve are playing game.
Alice and Eve are playing game.

Each player see different view according to his/her camera.
Alice and Eve are playing game.
Each player see different view according to his/her camera.

Alice’s entity

Eve’s entity

Walls
Alice and Eve are playing game. Each player see different view according to his/her camera.
Benign player

Alice’s View

Eve’s View

Alice’s entity

Walls

Eve’s entity
Benign player

*Benign* Eve won’t see Alice if she hides behind the wall
Wallhack!

Malicious Eve seeing through the wall with wallhack!
Benign player’s view
Wallhack view
Game client-server architecture

Alice’s machine

Client

Input

GPU

Alice

Server

Eve

Eve’s machine

Client

Input

GPU
Client-to-server

Client sends inputs to the server, and the server updates corresponding entity (e.g., Alice’s)
Server-to-client

Server sends the updates to the client
Server-to-client

Server sends the updates to the client even if the entity is not visible
Server-to-client

Server sends the updates to the client *even if the entity is not visible*
Server-to-client

Server sends the updates to the client even if the entity is not visible

Why? Performance & user experience issues
Attack surface 1: Client memory

An attacker probes client memory to find sensitive data (e.g., position)
Attack surface 1: Client memory

An attacker probes client memory to find sensitive data (e.g., position)

Anti-cheat tries to prevent attackers from reading client memory, but they are not perfect (arms race)
Attack surface 1: Client memory

An attacker probes client memory to find sensitive data (e.g., position)

Anti-cheat tries to prevent attackers from reading client memory, but they are not perfect (arms race)

Requirement 1: Contain sensitive data to secure region, where attackers cannot read
Rendering with GPU (Benign)

Client → Input

Eve’s machine

GPU

Vertex Processing → Rasterizer → Visibility Testing → Pixel Processing
Rendering with GPU (Benign)

Game state passed to GPU
Rendering with GPU (Benign)

Game state passed to GPU
Rendering with GPU (Benign)

Game state passed to GPU
Rendering with GPU (Benign)

Game state passed to GPU
Rendering with GPU (Benign)

Game state passed to GPU
Rendering with GPU (Benign)

Game state passed to GPU

Visibility testing discards invisible pixels
Rendering with GPU (Benign)

Game state passed to GPU

Visibility testing discards invisible pixels

Benign Eve won’t see Alice behind the wall
Attack surface 2: GPU

An attacker eavesdrops on CPU/GPU communication, or tamper with GPU computations.
An attacker eavesdrop on CPU/GPU communication, or tamper with GPU computations.
An attacker eavesdrop on CPU/GPU communication, or tamper with GPU computations
An attacker eavesdrop on CPU/GPU communication, or tamper with GPU computations.
Attack surface 2: GPU

An attacker eavesdrops on CPU/GPU communication, or tamper with GPU computations
Attack surface 2: GPU

Requirement 2: Prevent sensitive information from flowing into GPU

An attacker eavesdrop on CPU/GPU communication, or tamper with GPU computations
Summary of requirements
Summary of requirements

Requirement 3: Performance. Gamer want >60fps frame rate
Summary of requirements

Requirement 3: Performance. Gamer want >60fps frame rate

Requirement 4: No hardware modifications.
Summary of requirements

Requirement 1: Contain sensitive data to secure region, where attackers cannot read

Requirement 3: Performance. Gamer want >60fps frame rate

Requirement 4: No hardware modifications.
Summary of requirements

Requirement 1: Contain sensitive data to secure region, where attackers cannot read

Requirement 2: Prevent sensitive information from flowing into GPU

Requirement 3: Performance. Gamer want >60fps frame rate

Requirement 4: No hardware modifications.
Intel SGX

• TEE (Intel SGX) assumptions: confidentiality + integrity of enclave memory against privileged attackers (= cheaters)

• Widely available on commodity hardware
Intel SGX

• TEE (Intel SGX) assumptions: confidentiality + integrity of enclave memory against privileged attackers (= cheaters)

• Widely available on commodity hardware

Requirement 4: No hardware modifications.
Intel SGX

• TEE (Intel SGX) assumptions: confidentiality + integrity of enclave memory against privileged attackers (= cheaters)

Requirement 1: Contain sensitive data to secure region, where attackers cannot read

• Widely available on commodity hardware

Requirement 4: No hardware modifications.
Alternative designs
Alternative designs

CPU

client

Graphene SGX

GPU

Visibility testing
Alternative designs

Sensitive data passed to GPU (R2)
Alternative designs

Sensitive data passed to GPU (R2)
Alternative designs

Sensitive data passed to GPU (R2)

Requires hardware changes (R4)
Alternative designs

- **Graphene SGX**
  - CPU
  - Client
  - GPU
  - Visibility testing

- **Trusted GPU**
  - CPU
  - Client
  - Trusted GPU
  - Visibility testing

- **Software Renderer**
  - CPU
  - Client
  - Software Renderer

Sensitive data passed to GPU (R2)

Requires hardware changes (R4)
Alternative designs

- **CPU**
  - Client
  - Graphene SGX

- **GPU**
  - Visibility testing

Sensitive data passed to GPU (R2)

- **CPU**
  - Client
  - Graphene SGX

- **Trusted GPU**
  - Visibility testing

Requires hardware changes (R4)

- **CPU**
  - Client
  - **Software Renderer**
    - Graphene SGX

Prohibitive performance overhead (R3)
Alternative designs

- **CPU**
  - client
  - Graphene SGX

- **CPU**
  - client
  - Graphene SGX

- **CPU**
  - client
  - Software Renderer
  - Graphene SGX

- **CPU**
  - client
  - BlackMirror
  - Visibility testing

**GPU**

- Visibility testing

- Sensitive data passed to GPU (R2)

- Requires hardware changes (R4)

- Prohibitive performance overhead (R3)

**Our Approach**
BlackMirror: Trusted state

- BlackMirror stores the latest state of sensitive objects (*trusted state*)
  - Updates are received from a secure channel b/w the server and the enclave
  - *Local prediction* within the enclave (See paper)
- *Untrusted state* outside the enclave is used for rendering with GPU
BlackMirror: Trusted visibility testing

- **Trusted visibility testing** for each sensitive object in enclave
  - *Software renderer* inside the enclave constructs the *depth map* and tests each sensitive object
- Latest updates to invisible objects does not leave the enclave
  - Attacker only sees *stale* information of invisible objects
Enclave interfaces

• t_load_[world/model](): Load world and entity models into enclave
• t_parse_svc_secure(): Parse encrypted packets from the server
• t_predict_movement(): Predict movements with local inputs (See paper)
• t_test_packet_entities(): Build in-enclave depth map and test each entity against depth map. Only visible states are passed to untrusted state
• And more
Security Properties

- Any privileged software cannot access or modify enclave memory
- No sensitive state passed to the GPU (trusted visibility testing)
Depth map (enclave)
Benign player's view with BlackMirror
Benign player’s view with BlackMirror
Wallhack view with BlackMirror
Implementation & evaluation setup

• We prototyped BlackMirror with q2pro, Intel SGX SDK v2.7 and Intel MaskedOcclusionCulling for in-enclave visibility testing
• Intel i7-8700 (6-core), 16GB RAM, NVIDIA GeForce RTX 2080 Ti with 11GB GDDR 6
• Evaluation result shows BlackMirror running on a single thread
Evaluation

Latency to run a frame:

- BlackMirror: 1 enclave thread, S/W rendering -- 12 threads
- 60 fps \(\Leftrightarrow\) 16 ms per frame
- BlackMirror adds < 0.6 ms extra latency (including mode switching)
- NOTE: Trusted visibility testing doesn’t require all details

See paper for accuracy evaluation
Discussions & Limitations

• Aimbots
  • Currently BlackMirror leaves aimbots as out-of-scope

• Noticeability vs. Visibility
  • BlackMirror filtration mechanism relies on visibility (occlusion)
  • If an attacker try to improve the noticeability of partially occluded objects, say, by changing color of an entity, BlackMirror cannot prevent these types of attacks
Backup slides
Outline (XXX: To be updated)

• Wallhacks
• Why is it hard to stop wallhacks?
• BlackMirror
• Evaluation
• Discussions & Conclusion
BlackMirror: Wallhack prevention with TEE
BlackMirror: Wallhack prevention with TEE

Untrusted state is used to render the scene
BlackMirror: Wallhack prevention with TEE

BlackMirror stores trusted state (see paper for trusted state update and prediction)

Untrusted state is used to render the scene
BlackMirror: Wallhack prevention with TEE

Alice’s update is sent to BlackMirror via a secure channel

BlackMirror stores trusted state (see paper for trusted state update and prediction)

Untrusted state is used to render the scene
BlackMirror: Wallhack prevention with TEE

Alice’s update is sent to BlackMirror via a secure channel.

BlackMirror stores trusted state (see paper for trusted state update and prediction).

Trusted visibility testing determines which state is visible, and declassifies visible states.

Untrusted state is used to render the scene.