GPUによるメモリ書き換え監視を用いた高信頼アンチチートシステムの提案

橋本 直樹[1]　穐山 空道[1,a)]

概要：PC ゲームでのチート行為が問題となっており、対策としてカーネル空間で動作するアンチチートシステムが登場しているが、安全性の面で問題視されている。本研究では、カーネル空間で動作せず、かつ回避されづらいチート検知を行う方法として、GPU からゲームプロセスのヒープ内にあるゲームデータを監視をするシステムを提案する。GPU 上で実行中のプログラムに対するホストからの干渉が困難なことを利用し検知回避の難易度を向上させる。提案システムがヒープ内データを監視する時間間隔を計測したところ、4 byte のデータ 5,800 個を監視対象とした際に、1.67×10^{-2} 秒（$\approx \frac{1}{60}$ 秒）であることが分かった。

1. はじめに

PC ゲームで内部データの書き込みや読み込みを行いプレイを有利にするチート行為が問題となっている。特にオンラインゲームでのチート行為は他プレイヤーのゲーム体験に影響を与え、ゲーム開発会社に経済的損失をもたらす。チート行為に対抗するために様々なアンチチートシステムが開発されてきたが、その度にチートプログラムも形を変え、アンチチートシステムの監視を回避し続けてきた。

近年では、OS のカーネル空間で動作するアンチチートシステムが使用される。このシステムはカーネル空間に常駐し、PC 上で動く全てのプロセスを監視することで、多様なチートを検知し、チート開発の難易度を向上させている。しかし、このようにカーネル空間に外部のプログラムを注入し、大きな権限を持たせる行為は一般的に好まれていない。カーネルに常駐するアンチチートシステムの脆弱性を利用して、ランサムウェアなどの攻撃につながった事例もある [1]。また、同様にカーネル空間で動作するチートにより検知を回避されているのが現状である。

そこで本研究ではカーネル空間で動作せずかつ検知回避の難易度が高いアンチチートシステムの実現を目指す。

2. 提案システム

2.1 脅威モデル

本研究では以下の驚異モデルを仮定する。(1) 提案するシステムはゲームプロセス開始時に必ず正常に実行される。(2) 検知対象のチートは OS カーネル内で実行され、ユーザ空間に存在するゲームプロセスのヒープ内のデータを書き換える。(3) チートによって書き込まれる値は、通常のゲームプレイ上ではとりえない値である。例えば攻撃力を 999 にする、重力を 0 にするなどである。

[1] 立命館大学 情報理工学部
[a)] s-akym@fc.ritsumei.ac.jp

2.2 システムの概要

以上のようなチートを防止するため、GPU からホスト上のゲームプロセスのメモリ内容を監視しチートを検出するシステムを提案する。GPU は OS のユーザ、カーネルとは別のアドレス空間を持ち、OS 側から GPU メモリ内のプログラムを書き換えることは困難である。また、GPU には実行中のプログラムを任意に停止させる機能は用意されていない。以上から GPU 上で動作中のプログラムの動作を変更、停止させることは難しく、一度実行された本システムが悪意のある介入を受ける危険性は低い。

GPU からゲームプロセスのメモリへのアクセスは、ホストと GPU 間でマップされた空間への DMA (Direct Memory Access) を用いる。これは一度マッピングされるとホスト側での処理なしに GPU からのメモリアクセスが可能であり、チートプログラムによる介入を防ぐことができる。一方でユーザプロセスから GPU との通信 API を呼び出す方式では、API が攻撃者にフックされる危険性がある。

3. 設計と実装

提案システムの動作の流れを **図 1** に示す。ゲームプロセスは実行開始時に GPU で動作する監視プログラムを起動する。起動された監視プログラムは保護対象データの情報が登録されたテーブルを参照し、ゲームプロセスのヒー

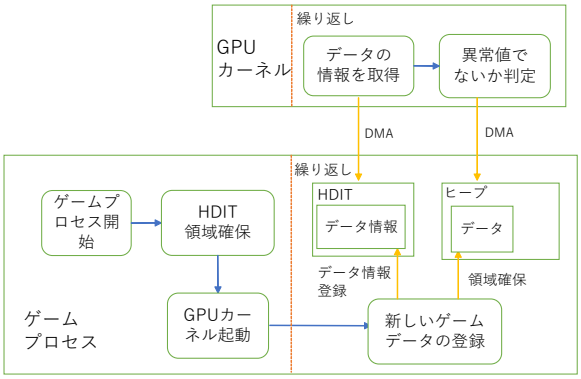

図 1: 本システムの構成と動作の流れ

表 1: 評価環境

GPU	NVIDIA GTX 1650
CPU	Intel Core i9 12900K
OS	Windows 11

プ内の保護対象データが異常な値でないかを監視する。

GPU からホストメモリへの DMA には、NVIDIA 社 の提供する `PageLockedMemory` 機能を使用する。ホストメモリへの DMA を行うには、アクセス対象の仮想ページが物理メモリ上に存在する必要がある。本機能は指定するメモリ範囲をスワップアウト不可に設定することでこの条件を常に成立させる。

GPU がホストのヒープ内のデータの仮想アドレスや正常な値の範囲を知るため、これらを HDIT (Heap Data Information Table) に格納する。HDIT はゲームプロセスがヒープに保護対象データを確保するたびに更新され、GPU は HDIT を参照し動的に新たなデータを監視対象に加える。具体的に HDIT には以下の情報を登録する。

(1) GPU カーネルの持つメモリ空間上で監視対象データにマップされたアドレス
(2) 保護対象データのサイズ
(3) 正常動作時に保護対象データが取りうる値の範囲
(4) ホストメモリ空間上の保護対象データのアドレス

ゲームプロセスは本システムが提供する新たなメモリ確保関数を利用し保護対象データを HDIT に登録する。この関数はメモリ領域の確保に加え、HDIT への情報の登録、`cudaHostRegister` 関数を用いた確保領域の `PageLockedMemory` への変換を行う。

4. 評価

本稿では提案システムが保護対象データを監視するのにかかる時間間隔を評価する。この間隔が短いほどオーバーヘッドが小さく、また短い間隔で有効無効を切り替えるようなチートを発見しやすいと言える。具体的には HDIT の参照と保護対象データの参照及びチートの判定を 1 サイクルとして、1 サイクルの開始から終了までの時間を計測する。計測に使用した環境を **表 1** に示す。

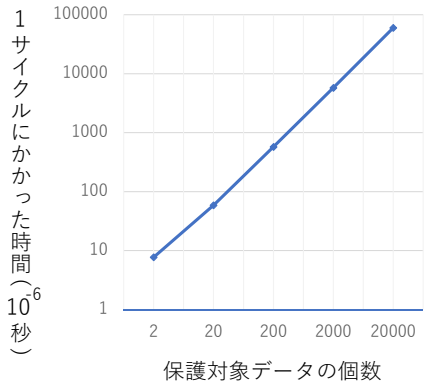

図 2: 1 サイクルにかかった時間とデータ数の対応

図 2 に 4 byte の保護対象データの個数と 1 サイクルにかかった時間の関係を示す。図よりデータの個数と 1 サイクルにかかった時間は傾き 1 でほぼ比例する。例えば 60 fps のゲームの 1 フレームは 1.67×10^{-2} 秒 ($\approx \frac{1}{60}$ 秒) であり、本システムで 1 フレームの間だけゲームデータを書き換えるチートを確実に検知するには 4 byte の保護対象データを 約 5,800 個以内に収める必要があると分かる。

5. 関連研究と今後の課題

関連研究として GPU によるシステム障害検知がある [2]。これは GPU から OS の管理データを監視することで OS の障害を検知するシステムである。本研究はゲームのチートを監視する点、実行時に動的に生成される監視対象データの追跡のために HDIT を持つ点が大きく異なる。

今後の課題は以下である。第一に、今回検知したもの以外のメモリ書き換えチートへの対応が必要である。第二に、GPU 上の複数コアでヒープを分割スキャンすることが検知性能向上に有効である。また UPMEM [3] のような CPU から独立した計算能力とメモリアクセス能力を持つデバイスで同様の監視を行うことも考えられる。

謝辞 本研究は JSPS 科研費 JP22H03566 の助成を受けたものです。

参考文献

[1] Soliven, R. and Kimura, H.: Ransomware Actor Abuses Genshin Impact Anti-Cheat Driver to Kill Antivirus, `https://www.trendmicro.com/en_za/research/22/h/ransomware-actor-abuses-genshin-impact-anti-cheat-driver-to-kill-antivirus`.
[2] Ozaki, Y., Kanamoto, S., Yamamoto, H. and Kourai, K.: Detecting System Failures with GPUs and LLVM, *APSys*, p. 47–53 (2019).
[3] Nider, J., Mustard, C., Zoltan, A., Ramsden, J., Liu, L., Grossbard, J., Dashti, M., Jodin, R., Ghiti, A., Chauzi, J. and Fedorova, A.: A Case Study of Processing-in-Memory in off-the-Shelf Systems, *USENIX ATC*, pp. 117–130 (2021).