Windows 用アンチチートツールのコールバック改ざん防止機能の Linux への移植方法の検討

稲森 大翔[1]　穐山 空道[1,a)]

概要：近年の PC ゲームはカーネル空間で動作するアンチチートシステムを要求し、このためある OS 用に開発されたゲームを他 OS でプレイすることが難しくなっている。OS が提供する機能は Os ごとに違うため、カーネル空間で動作するアンチチートツールを単純に移植することは難しい。そこで本研究では、カーネル空間で動作するアンチチートツールを他 OS に移植方法を検討する。具体的には原神で使用されている Windows 用のアンチチートツールである mhyprot2 の機能を Linux で再現する方法を検討する。

1. はじめに

近年のオンラインゲームではカーネル空間上でのチート対策が求められており、様々な企業がカーネル空間で動作するアンチチートツールを開発している。例えば人気ゲームを開発元である Riot Games 社の vanguard、大手ゲームプラフォーム steam の開発元である Valve Corporation の VAC (Valve Anti-Cheat)、また今回取り上げる miHoYo 社の mhyprot2 などがある。

カーネル空間で動作するアンチチートツールを異種 OS 間で移植することは難しい。機能の移植を達成するためにはその機能が利用している OS API と同等の機能が移植先の OS でも提供されている必要がある。しかし OS が違うと OS の提供する機能が全く違うため、コードそのものから変える必要があり新しく作り直すことになってしまう。

アンチチートツールの移植が難しいことは、ゲーム自体の互換性に悪影響を与えている。近年、Valve 社の互換レイヤー Proton のおかげで異種 OS 間でコードの変更なし遊べるゲームが増えてきてはいるが、steam の最低基準の人気度を満たすゲーム 9,000 タイトルのうち、Linux で実際にプレイ可能なゲームは 16 % しかない。

そこで本研究では、Windows のカーネル空間で動作するアンチチートツールを Linux に移植する方法を検討する。

2. 調査

2.1 mhyprot2 のアンチチート機能

アンチチートツールを移植するためにはそれぞれの環境で機能を再現する必要がある。本研究では、原神で実際に使用されているアンチチートツール mhyprot2 の特定の機能を Linux に移植できるかについて考える。

具体的に、mhyprot2 の機能のうち「チートツールによって mhyprot2 自身が改変されないかを確認する機能」を考える。この機能の実現には以下の 3 つのカーネル API と 1 つのリストが使用されている [1]。

(1) PsSetCreateProcessNotifyRoutine 関数
(2) PsSetCreateThreadNotifyRoutine 関数
(3) PsSetLoadImageNotifyRoutine 関数
(4) PspLoadImageNotifyRoutine リスト

PsSetCreateProcessNotifyRoutine 関数はプロセスが作成、削除された際に呼ばれるコールバックを登録する。PsSetCreateThreadNotifyRoutine 関数も同様にスレッドが作成、削除された際に呼ばれるコールバックを登録する。PsSetLoadImageNotifyRoutine 関数は dll などが読み込まれメモリにマップされる際に呼ばれるコールバックを登録する。PspLoadImageNotifyRoutine 配列は登録されたコールバック関数へのポインタが格納される最大 64 個のリストである。

これらを利用し、mhyprot2 は **図 1** のように改変を検知する。自身が持つ登録されているはずのコールバック関数のポインタと、Windows が持つ実際に登録されているコールバック関数のポインタを比較し、登録されているはずのコールバックが削除されていないかを確認する。

2.2 Linux での再現

mhyprot2 の改変検証機能が Linux で再現可能かについて調査する。改変検証機能に必要な機能は大きく分けて二つある。

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

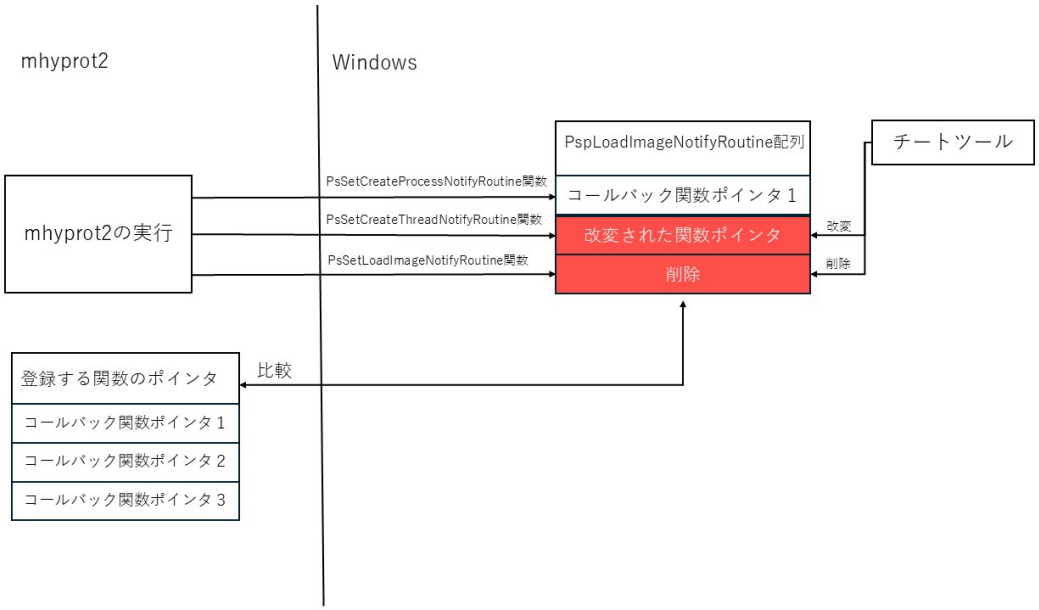

図 1: mhyprot2 の耐改変機能の仕組み

(1) PsCreateProcessNotifyRoutine 関数のようなコールバック関数を登録する機能
(2) 登録されているはずのコールバック関数のポインタを取得する機能

一つ目の機能は Linux の ftrace [2] を使用することで再現できる。ftrace は Linux カーネル内の様々な関数の実行をトレースする機能であり、カーネル内の関数名を指定すると指定した関数が呼び出された際にコールバック関数が呼び出される。ftrace を用い、`__x64_sys_clone` をトレース対象として指定することでプロセスが作成された際にコールバック関数を呼ぶことができる。

二つ目の機能の実現には、ftrace で登録されたコールバック関数へのポインタのリストが必要である。本研究ではこの実現可能性を探るため、Linux v6.11 のソースコードを調査する。ftrace の実装である `ftrace.c` を見ると、`set_ftrace_ops_ro` 関数の中で `ftrace_ops_list` というリストを走査している。このリストは `ftrace_ops` 構造体のリストであり、本構造体がコールバック関数の情報を保持している。従ってこのリストを走査れば登録済みのコールバック関数の一覧が取得できると考えられる。

3. 関連研究

伊藤らは、組込み OS 向けのドライバを開発する際の開発の負担を削減するため Linux 用のドライバを移植することを提案する [3]。移植するために Linux 用のドライバが依存している Linux カーネル内のシンボルを自動的に抽出する。Windows でも同様のことができれば本研究の目的に役立つ可能性があるが、Windows ではソースコードが存在しないためそのままの適用は難しい。

また、ndiswrapper [4] は、Linux 上で Windows 用の Wi-Fi ドライバを改変なしで動作可能にする。NDIS (Network Driver Interface Specification) と呼ばれるネットワークドライバ用の共通 API を実装することで、少ない互換レイヤの実装労力でこれを可能にする。

4. 課題

この研究の今後の課題として大きく分けて二つと考える。一つ目はあくまで今回の機能の再現は mhyprot2 が持つうちの一つの機能であり、他のアンチチートツールも同様の方法で再現できるかどうか不明である点である。はじめにで挙げた通りカーネル内のアンチチートツールにも種類があり、さらにそれぞれ違う機能を持つため全てを Linux で再現できるかどうかは未知である。二つ目は今回の移植方法は手動で再現できる機能をコードから探し出す方法をとっているため、自動で移植ができていない点である。アンチチートツールはチートツールの新たな攻撃、その攻撃の対策の繰り返しなため常に更新され続ける。そのため更新されるたびに手動で移植していると効率が悪い、自動化することが望まれるがその方法が必要である。

参考文献

[1] meekochii: Analyzing Genshin Impact's Anticheat Module, `https://research.meekolab.com/analyzing-genshin-impacts-anticheat-module` (2022).
[2] The kernel development community: Using ftrace to hook to functions, `https://www.kernel.org/doc/html/v6.11/trace/ftrace-uses.html`.
[3] 伊藤大智, 中野颯, 井内晴菜, 福田泰平, 毛利公一: シンボル情報の解析によるイーサネットデバイスドライバと Linux kernel の依存調査, 研究報告システムソフトウェアとオペレーティング・システム (OS), pp. 1–7 (2024).
[4] Brinley: Main Page, `https://ndiswrapper.sourceforge.net/wiki/index.php/Main_Page` (2010).