Windows OS上でのフィルタドライバを用いたセキュアアクセス制御機構の構築

†安藤類央　　‡井上宣子　　†須崎 有康

†情報通信研究機構
〒184-8795 東京都小金井市貫井北町 4-2-1

‡サイエンスパーク株式会社
〒252-0816 神奈川県藤沢市遠藤５３２２

†産業技術総合研究所
〒101-0021 東京都千代田区外神田 1-18-13

あらまし　本論文では、Windows OS上でのファイルシステムフィルタドライバを用いたセキュアアクセス制御機構の構築手法について提案する。デバイスドライバを用いることで、Windows OSのようなソースが公開されていないシステムに関しても、セキュアＯＳのアクセス制御機構を構築することが可能になる。提案システムでは、ファイルシステムでのインターセプト、フィルタリングに最適化されたフィルタマネージャを用いて、アクセス時刻、プロセス、アクセスオブジェクトなどの要素によるセキュアＯＳのアクセス制御機構を構築し、ゼロデイ攻撃などのセキュリティインシデントの被害を最小化する。
キーワード：セキュアアクセス制御、ファイルシステムドライバ、フィルタマネージャ、Windows OS、ゼロデイ攻撃

An implementation of secure access control on Windows OS using filter driver

†Ando Ruo　　‡Takahashi Kazushi　　‡Kuniyasu Suzaki

†National Institute of Information and Communications Technology,
4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795 Japan
†SciencePark Corporation
1-1538-11 Iriya Zama, Kanagawa, Japan 228-0024
‡11F Akihabara Dai Bldg., 1-18-13 Sotokanda Chiyoda-ku,
Tokyo 101-0021 Japan

Abstract　With the rapid advances of the developing environment of device driver, secure access control is becoming possible to some extent even in closed source system sush as Windows OS. In this paper we propose an implementation method of secure access control mainly focused on file system of Windows OS using file system filter driver. In proposed system, we apply a filter mananger, newly introduced filtering system of file IO to intercept APIs to trace the access time, process and access object to achieve secure access control. With our modules, security incident on Windows OS could be prevented and minimized using file access filtering.
Keywords: Secure access control, file system driver, filter manager, Windows OS, Zero-day attacks

1　はじめに

近年のデバッグ機構に関する急速な需要の増大と関連技術の発達は、Windows OS などのクローズドソース（閉鎖系）システムの修正と操作をオープンソース（開放系）と同程度に可能にしたといえる。従来 Linux などのオープンソース OS でのみ実装が行われていたセキュア OS アクセス制御が、Windows OS でもある程度可能になってきている。本論文では、Windows OS 上でのフィルタドライバを用いたセキュアアクセス制御機構の構築法を提案する。

2　提案システム

フィルタドライバは、フィルタドライバは、ＩＯマネージャとカーネルドライバの間に位置して、通過する IRQ を修整することでファンクションドライバ（既存のデバイスドライバ）に任意の機能を追加することができる。任意の関数内に、発行（アクセス元）プロセス ID、アクセスオブジェクト、IO 要求の種別などの情報を獲得することが可能であり、これらの情報を利用してセキュア OS が採用しているアクセス制御をある程度、実現することが可能である。図１に、フィルタマネージャの稼動位置とファイルアクセス制御の概要を示した。提案モジュールはファイルアクセスを行うアプリケーションとファイルシステム・ディスクドライバの間に位置し、設定されたアクセスリストによって IO 要求を通過あるいは修整破棄することでアクセス制御を行う。

3　適用技術

Window OS の修正は、Microsoft が提供する API をインターセプトすることで可能である。API の修正技術には、ユーザモードでは、DLL Injection、カーネルモードでは API フックと、フィルタマネージャの利用がある。提案システムの構築では、フィルタドライバと用いた API フックと、フィルタマネージャを用いた。

3.1　フィルターマネージャ

開発環境や稼動する状況にも依存するが、通常のフィルタドライバを用いたファイルＩＯのフックは、安定動作しない場合がある。フィルタドライバは、Microsoft が新たに提供をはじめたファイルシステムフィルタドライバである。フィルタドライバは、大量に呼び出される割り込みやＡＰＩに関して、順序などを適切に制御し、サードパーティの開発を簡素化、支援することを主眼に設計されている。図２は、フィルタマネージャの機能概要を示したものである。フィルタマネージャは、カーネルドライバ（ファンクションドライバ）とフィルタドライバの間を仲介し、フィルタドライバの実装を簡素化し、機能を安定にする。

3.2　フィルタドライバ

Microsoft Windows のフィルタドライバは、ＸＰから積極的に導入、活用が始まったソフトウェアモジュールである。フィルタドライバは、ＩＯマネージャとカーネルドライバの間に位置して、ファンクションドライバ（既存のデバイスドライバ）の前後に既存の Windows が提供する機能を利用して呼び出され、新しい機能を追加したり、機能修正、デバッグなどを行うソフトウェアである。フィルタドライバを用いることで、Native API フックを行うことができる。Native API Hook は、カーネルモードでのイベント検出に用いられる。ユーザモードでの API 発行は、ntdll.dll 経由でカーネル空間に伝達される。Windows では、カーネルモードで動作する API を、native API といい、これらは、SystemServiceDescriptorTable という構造体によって、制御されている。そのため、Native API Hook を行うためには、SystemServiceDescriptor に修正を加える必要がある。

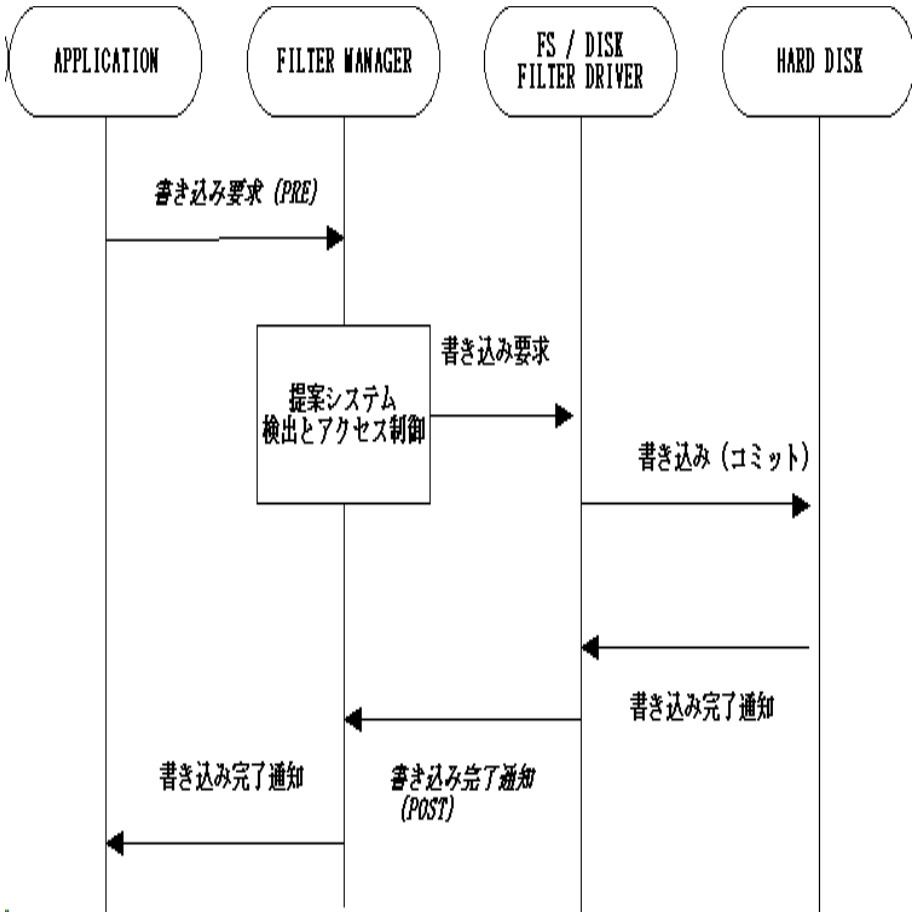

図 1: フィルタマネージャの稼動位置とファイルアクセス制御の概要

4 適用アクセス制御

4.1 MAC (Mandaroty Access Control)

MAC (Mandatory Access Control) は、セキュア OS の代表的なアクセス制御の 1 つである。MAC の背景概念として DAC(Discretionary Access Control) がある。DAC：任意アクセス制御は、アクセス制御（ファイルのパーミッションなど）はメカニズムの利用者が決定できるため、exploit によって権限を奪われたユーザによって変更されたアクセス権から情報が漏洩してしまう可能性がある。これに対し、MAC はセキュリティポリシーを設定するファイルによってアクセス権を制御管理する。DAC は機構のユーザがアクセス制御を行うのに対し、MAC はメカニズム自体がアクセス制御を行う。提案手法では、Windows OS に組み込まれるドライバがアクセス制御を行うことにより、セキュアアクセス制御機構の構築を行う。

4.2 TDE (Trusted Domain Enforcement)

悪意のある実行ファイルへのセキュリティ確保の手法として、TPE (Trusted Path Execution) と TDE (Trusted Domain Execution) がある。TPE は、外部から持ち込まれ生成された実行ファイルに対して有効であるが、シェルやその他の実行処理系が信頼された場合、悪意のある入力による動作を検査することができない。TDE は、実装処理系をサンドボックス化する、あるいは保護されてない入力があったときに実行処

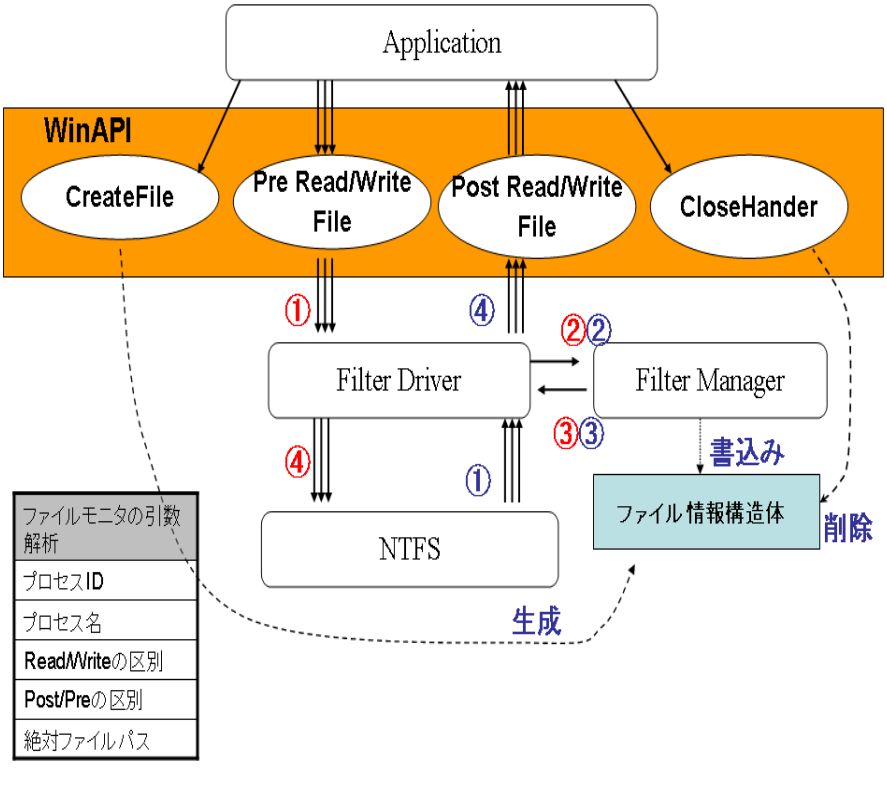

図 2: ファイルＩＯのイベントを検出、修正するためのフィルタマネージャの機構の概略

理系のケーパビリティを変更するというものである。

5　フィルタマネージャの詳細

OSがドライバをロードすると、IOマネージャからDriverEntryが起動される。フィルタマネージャ機構において、DriverEntryはミニフィルタドライバとして登録を行い、フィルタリングを開始する。ミニフィルタドライバとして登録を行う際には、FLT_REGISTRATION構造体を利用する。各IRPに対応するコールバックルーチンをFLT_REGISTRATION構造体のOperationRegistrationに設定する。ファイルシステムがロードするとFLT_REGISTRATION構造体のInstanceSetupCallBackに設定したコールバック関数が呼び出される。表１に、提案モジュールでサポートするIRQ要求を示す。表２、３に、IRP_MJ_CREATE(PRE)、IRP_MJ_CREATE(POST)のディスパッチルーチンの概要を示す。

6　まとめと今後の課題

本論文では、Windows OS上でのファイルシステムフィルタドライバを用いたセキュアアクセス制御機構の構築手法について提案する。デバイスドライバを用いることで、Windows OSのようなソースが公開されていないシステムに関しても、セキュアＯＳのアクセス制御機構を構築することが可能になる。提案システムでは、ファイルシステムでのインターセプト、フィル

No	IRQ 要求	PRE	POST
1	IRQ_MJ_CREATE	○	○
2	IRQ_MJ_CLOSE	○	×
3	IRQ_MJ_CLEANUP	○	×
4	IRQ_MJ_READ	○	×
5	IRQ_MJ_WRITE	○	×
6	IRQ_MJ_SET_INFORMATION	○	×

表 1: 提案システムでサポートする IRQ 要求

ディスパッチルーチン	Common ディスパッチ
関数の型	FLT_PREOP_CALLBACK_STATUS
機能	ファイル名、プロセス名を取得する。
引数 1	PFLT_CALLBACK_DATA Data
引数 2	PCFLT_RELATED_OBJECTS FltObjects
引数 3	PVOID *CompletionContext
戻り値 1	FLT_PREOP_SUCCESS_WITH_CALLBACK で正常終了
戻り値 2	FLT_PREOP_SUCCESS_WITH_CALLBACK 以外で異常終了

表 2: IRP_MJ_CREATE(PRE) アプリケーションのオープン要求をファイルシステムに発行する前に呼び出される。

タリングに最適化されたフィルタマネージャを用いて、アクセス時刻、プロセス、アクセスオブジェクトなどの要素によるセキュアＯＳのアクセス制御機構を構築し、ゼロデイ攻撃などのセキュリティインシデントの被害を最小化する。

参考文献

[1] File System Filter Drivers, Microsoft http://www.microsoft.com/whdc/

[2] Windows Hardware Engineering Conference http://www.microsoft.com/whdc/winhec

[3] Programming Applications for Microsoft Windows Forth Edition, Jeffrey Ritcher, Microsoft Press, 1999

[4] Gary Nebbett, Windows NT/2000 Native API Reference, Sams Publishing; ISBN:1578701996,2000

[5] Kosoresow, Andrew P. and Steven A. Hofmeyr, "Intrusion Detection Via System Call Traces", IEEE Software, Sept..Oct. 1997, pp 35-40.

[6] Yusuf Wilajati Purna, "LIDS Trusted Domain Enforcement", 2004 available at: http://www.lids.org/

[7] Niki A. Rahimi, "Trusted path execution for the linux 2.6 kernel as a linux security module", In Proceedings of the USENIX Annual Technical Conference 2004.

[8] Ruo Ando, Youki Kadobayashi, "Improving VMM based IPS for real-time snapshot and nullification of buffer overflow exploitation" The 1st Joint Workshop on Information Security September, 20-21, Sookmyung Women's University, Korea

[9] Trusted Computer Systems Evaluation Criteria

ディスパッチルーチン	Common ディスパッチ
関数の型	FLT_POST_CALLBACK_STATUS
機能	ファイル毎に保持するコンテキストを取得する。
引数 1	PFLT_CALLBACK_DATA Data
引数 2	PCFLT_RELATED_OBJECTS FltObjects
引数 3	PVOID *CompletionContext
引数 4	FLT_POST_OPERATION_FLAGS Flags
戻り値 1	FLT_POSTOP_FINISHED_PROCESSING で正常終了
戻り値 2	FLT_POSTOP_SUCCESS_WITH_CALLBACK 以外で異常終了

表 3: IRP_MJ_CREATE(POST)　アプリケーションのオープン要求をファイルシステムに発行した後に呼び出される。

http://www.auditmypc.com/
acronym/TCSEC.asp

[10] Chris M Wright, Linux Security Module Framework, Linux Symposium 2002.

[11] 安藤類央, 野崎隆, 武藤佳恭、「ファイルシステムドライバを用いたメタモーフィックコーディングの検出」, 情報処理学会 コンピュータセキュリティ研究会 CSS2004 研究報告, 2004 年 10 月 20 日

[12] 安藤類央、Nguyen Anh Quynh、須崎有康、「Windows のメモリ挙動モニタと Libvirt によるゼロデイ攻撃の検出システムの構築」, 暗号と情報セキュリティシンポジウム（SCIS2009）2009 年 1 月 21 日

[13] Windows Driver Kit
http://www.microsoft.com/whdc/devtools/
WDK/default.mspx