Vol.2021-DPS-186 No.48
Vol.2021-CSEC-92 No.48
2021/3/16

ユーザ操作特定のためのカーネル内でのプロセス挙動収集手法

藤枝 慶弘[1,a)]　羽角 太地[2]　島 成佳[2]　安田 真悟[3]　鄭 俊俊[1]　毛利 公一[1]

概要：標的型攻撃で侵入してくる攻撃者の継続的な観測を行うための環境を構築するためには，侵入先の計算機がユーザによって普段から利用されていることを模擬する必要がある．このような観測環境を効率的に構築するために，実際のユーザによる操作列に関する情報を記録し，それを元に再現することを検討している．本論文では，ユーザのアプリケーション操作を再現するために必要な情報を取得することを目的として，具体的には Windows におけるカーネルモードでプロセス挙動を取得するソフトウェアを設計・実装・評価したので報告する．ユーザがローカルシステム上でログオンした際に生成されるセッション情報，資格情報，アプリケーションの実行やファイルの操作などのイベントなど，プロセスやスレッドの情報をカーネルモードで収集する．これにより，取得した情報から継続的なユーザの操作を特定できることを確認した．

キーワード：標的型攻撃，カーネルモード，オペレーティングシステム

Method for collecting process behavior in the Kernel to identify User Operations

Abstract: In order to construct an environment for continuous observation of an attacker in a targeted attack, it is necessary to simulate the usual use of the target computer by users. In order to construct such an observation environment efficiently, we are considering recording information about the sequence of operations by actual users and reproducing them based on the recorded information. In this paper, we report on the design, implementation, and evaluation of a software program that obtains information necessary to reproduce the user's application operations, specifically, the process behavior in kernel mode in Windows. The software collects process and thread information in kernel mode, such as session information generated when a user logs on to the local system, token information, and events such as application execution and file operations. We have confirmed that we can identify continuous user operations from the acquired information.

Keywords: Advanced Persistent Threat，Kernel Mode，Operating System

1. はじめに

サイバー犯罪の検挙数は年々増加傾向にあり [1]，特に特定の組織の人間を対象として攻撃を行う標的型攻撃は，攻撃対象毎にその環境に合わせた攻撃手口へと変えるため，被害が多く問題となっている．このような標的型攻撃では，攻撃傾向を明らかにし，企業などの組織に対して注意喚起を行うことで被害を防ぐことができる．そのため，サイバー攻撃誘因基盤 STARDUST[2] のような攻撃者の活動を継続的に観測可能な環境を用いて，攻撃の活動によって観測環境に残る記録を収集する必要がある．

標的型攻撃の観測環境では，実際の組織を模したネットワーク構成や計算機を仮想的に用意して攻撃者を誘い込み，プロキシや攻撃観測システムによって挙動を観測する．

[1] 立命館大学
Ritsumeikan University
[2] 日本電気株式会社
NEC Corporation
[3] 国立研究開発法人情報通信研究機構
National Institute of Information and Communications Technology
[a)] yfujieda@asl.cs.ritsumei.ac.jp

しかし，このような環境では攻撃者が観測環境であること検知することによって攻撃活動を中止してしまい，攻撃の解析や研究活動に十分な記録が残らない場合があるといった課題がある．そのため，このような観測環境では侵入先の計算機がユーザによって普段から利用されていることを模擬し，攻撃者に自然な環境であることを認識させることで興味関心を引き，継続的な攻撃の観測を実現する必要がある．

一般的にユーザが計算機を利用する場合，計算機に対してログオンし，アプリケーションを利用し，活動が終了すると計算機からログアウトする．こうしたログオンからログオフまでの活動の流れでは，ユーザが利用するアプリケーションの他システム関連のプロセスなども起動されるため，計算機の記録ではユーザによって起動されたアプリケーションの記録を識別することが難しい．そのため，計算機上での特定のユーザの活動記録を取得するためには，ユーザのログオンからログオフまでの活動を計算機上の記録から分離する必要がある．

以上の背景から，本論文では計算機上の記録がどのユーザのものか特定するために，ユーザによるログオンの情報を含めた記録を取得する手法を提案する．提案手法は，Windows を対象としてユーザによって生成されたプロセスにはユーザのログオン情報が含まれることに着目し，ユーザのログオンからログオフまでに生成，終了されたプロセスの記録を収集する．また，ユーザがプロセスを通じて行う操作の一例としてファイルの変更を対象としてファイルの変更がどのプロセスやスレッドに行われたかを判別する情報の収集も行う．

以下，本論文では，2 章で関連研究について述べ，3 章で提案手法について述べる．4 章で提案手法の実装について述べ，5 章で評価について述べ，6 章でまとめる．

2. 関連研究

計算機上での活動の記録を残す手法として，これまで多くのシステムが提案されている．本章では，既存のを収集する手法について述べる．

竹久ら [3] は，カーネルモードドライバにてプロセス生成やスレッド生成のカーネル API をフックすることでプロセスの挙動を網羅的に記録として残す手法を提案している．この研究では，Windows のカーネルモードで動作するデバイスドライバだけでプロセス情報を収集し外部への送信を行うシステムの研究開発を行っている．しかし，このシステムでは作成者のユーザやユーザの資格情報を明らかにする情報を残していない．そのため，ユーザの操作と収集した情報を関連付けることが難しい．

Sysmon[4] は，様々なシステムアクティビティを記録し，イベントログとして出力を行う．プロセス生成やファイル変更ネットワーク接続などプロセスの挙動を記録可能である．しかし，Sysmon は前述の作成者情報が記録されていないことに加えて，プロセス生成やネットワークトラフィックの観測などの膨大な記録を残すイベントについて出力する情報量を制限することができないため，イベントログの容量を圧迫し記録可能な容量を超える可能性がある．

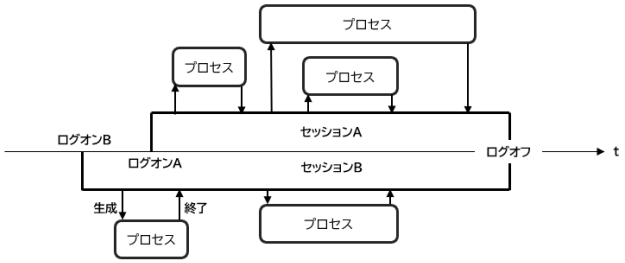

図 1 提案手法の全体像

以上のように，プロセスの挙動における操作者の情報や作成者の情報を記録する必要がある．また，リソースの圧迫を最低限にするため，ユーザが行った操作ではない記録については出力を制限する必要がある．

3. 提案手法

3.1 概要

Windows では，ログオンを行いログオフまでにユーザが活動を行うための空間をセッションと呼ぶ．システムを含めた複数のユーザがセッションを作成し，並行して操作が行われるためどのユーザによって生成されたプロセスかを判別するこが難しい．しかし，計算機上で記録される複数ユーザの活動をセッション毎に分離することで特定のユーザによって生成されたプロセスを特定することができる．ユーザは，計算機の操作を行う際にログオンを行い，ログオフまで当該セッション内でプロセスを通じてファイルの変更をはじめとした活動を行う．そのため，セッション毎に動作するプロセスを特定することプロセスが行った活動からユーザに起因するものを関連付けることができる．

図 1 に，セッションを用いたプロセスの分離する方法を示す．セッションA，Bはそれぞれ別のユーザによってログオンされたものである．セッション A で生成されたプロセスはセッション A の情報を用いることでセッションBのプロセスと分離することができる．

そこで，以下のイベントを対象として，これらのユーザ操作における操作したユーザを特定可能な情報を記録する手法を提案する．

- ログオンイベント
- プロセス生成イベント
- ファイル変更イベント

提案手法では，記録対象を網羅的に収集するためカーネルモードで実装を行い，ユーザ操作が行われた際にフックを行い，記録に必要なカーネルオブジェクト内の情報を収

集する．以下，本章では提案手法の構成要素における要件について述べる．

3.2 ログオンイベントの記録

ユーザによるログオン，ローカルでのログオンやリモートでのログオンなどログオン時のユーザの状態を区別するため，ログオンイベント発生時にセッション情報の収集を行う．ユーザがシステムにログオンするにはリモートでのログオンやローカルでのログオンが考えられるため，どのようなログオンを行ったかを区別する必要がある．また，継続的な観測においてユーザによるログオンがどのタイミングで行われていたを区別する必要がある．以上を判別可能とするため，以下の情報を提案手法では収集する．

- セッション ID
- ログオン ID
- ログオンユーザ名
- ログオンユーザのドメイン名

I/O マネージャの通知機能を用いてログオンフックを行いセッション ID の取得を行う．また，ログオンを行ったユーザの情報をログオンシーケンスの追跡を行い特定する．

3.3 プロセス生成イベントの記録

ユーザモードにおけるプロセスは Windows においてユーザがログオンしたログオンセッションによって分離される．そのため，ユーザモードでプロセス生成イベントを観測する場合，観測元ログオンセッション以外のセッションで生成されたプロセスの生成を観測することができない．そこで，プロセス生成イベントの記録をカーネルモードで処理を行う．

プロセスに割り当てられるトークンには，オブジェクトに対するアクセス権を示す情報であるトークンが管理されている．そのため，トークンの情報をプロセスオブジェクトから取得することで，フォレンジック時にユーザによって生成されたプロセスがどのようなアクセス権を保有していたかを特定可能な情報を取得する．また，プロセスの作成者を特定するためには，ユーザ名やドメイン名のようなユーザ情報に加え，いつログオンしたユーザか特定しなければならない．以上の内容から，以下の情報を収集することとする．

- 作成元セッション ID
- 作成元ログオン ID
- 作成ユーザ名
- 作成ユーザのドメイン名
- トークン ID
- アプリケーションタイプ

3.4 ファイル変更イベントの記録

ファイルの変更はカーネルコンポーネントの I/O マネージャを通じて行われ，ユーザ空間で観測することは困難である．I/O マネージャを含む I/O コンポーネントの集合である I/O システムは，アンチウイルスなどのファイルの I/O 処理をフックする必要があるシステムに向けて，File System Mini Filter Driver が用意されている．このカーネルモードドライバでは，ファイルシステムドライバへの処理を実行する前にファイルの I/O 処理をフック可能であるため，File System Minifilter Driver を用いてファイル操作のフックを行う．

ファイル操作におけるユーザによる GUI 操作の判定を行うためには，作成元のスレッドが GUI スレッドであるか知る必要がある．また，作成元のスレッドが動作するプロセスがユーザによって生成されたものか分からなければならない．以上の内容から，以下の情報を収集することとする．

- 作成元プロセス ID
- 作成元スレッド ID
- 作成元スレッドの GUI フラグ
- ファイルパス
- ファイルの変更操作

4. 実装

4.1 ログオンイベント

提案手法におけるログオンイベントにおけるセッション情報の記録実現するため，カーネルモードドライバとサービスを用いてセッションイベント，ログオンイベントのフックを行い，セッション情報の収集機能の実装を行った．本章では，ログオンイベントのフック方法が存在しないため Windows におけるログオンシーケンスの調査結果を用いてログオンイベントを特定しセッション情報とログオン情報の収集機能の実装を行った．

4.1.1 ログオンのフック

カーネルモードでは IoRegisterContainerNotification 関数を用いることでログオンイベントの通知を受け取ることができる．この関数にてログオンイベントの発生通知を受けることでログオン時に生成されたセッションオブジェクトを収集することができる．セッションオブジェクトはターミナルセッションのカーネルオブジェクトであり，カーネル内でターミナルセッションを管理する役割がある．しかし，セッションオブジェクトの構成は公開されておらず，セッションオブジェクトから直接セッション ID を取得することができない．セッションオブジェクトには，バージョンによってオフセットの差異がある可能性はあるが MM_SESSION_SPACE 構造体が含まれる．そこで，セッションオブジェクトから MM_SESSION_SPACE 構造体を辿ることでセッション ID の取得を行った．この方法では，セッションＩＤとセッションの正確な生成タイミングを取得可能だが，セッションの作成者情報の収集を行うことが

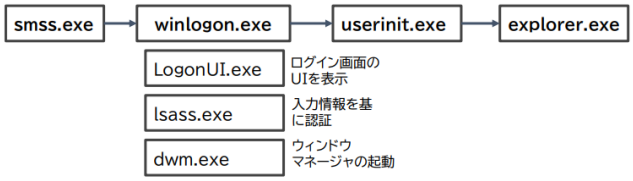

図 2 Windows におけるログオンシーケンス

できないため，ログオンシーケンスの観測を行うことで作成者情報の収集を行う．

4.1.2 ログオンシーケンスの調査

Windows では，端末を仮想的に分離しターミナルサービスによってリモートデスクトップやローカルでの複数の端末操作を可能にするためセッションといった概念が取り入れられている．セッションは仮想端末への接続が行われると作成され，仮想端末毎に固有のセッションが割り当てられる．また，仮想端末内に複数のユーザがログオンすることを可能にするため，ログオンセッションといった上記のセッションとは別の概念も利用されている．サービスや OS のサブシステムはシステムセッションでプロセスが生成され，予約されているセッション 0 を利用する．ユーザが生成したプロセスにはセッション 1 以降のセッションが利用される．そのため，セッションとログオンセッションの情報を取得することで，ユーザがどの端末でコンピュータを利用しているか，どのユーザがプロセスを生成したかを解析することができる．図 2 に，Windows が起動されユーザがログオンしログオンシェルである explorer.exe が起動されるまでの流れを示す．

Windows が起動されるとまず ntoskrnl.exe がロードされ起動に必要なドライバやレジストリの読み込みが行われる．上記の処理が終了した後，図 2 で示す処理が始まり，まず smss.exe が起動される．smss.exe はセッションの初期化，上記の処理で読み込まれなかったドライバやレジストリの読み込みが行われ，ログオン処理を行う winlogon.exe が起動される．winlogon.exe はログイン画面の表示や認証，ウィンドウマネージャを行うサブシステムを立ち上げログオン処理が終了した後にログオンセッションの生成を行う userinit.exe の起動を行う．userinit.exe はログオンセッションの生成が終了し，ユーザのログオンが可能になるとログオンシェルである explorer.exe を起動する．

上記の処理の流れからユーザによるログオンの成立はログオンシェルである explorer.exe が起動されたタイミングである．そこで，本提案手法ではユーザによるログオンイベントの特定を，上記の一連の処理を観測することで実現する．

4.1.3 ログオン情報の収集

ログオンシーケンスの調査によって userinit.exe によるログオンシェル explorer.exe が起動されたことがユーザによってログオンが行われたタイミングであることが明らかになった．そこで，ログオンシーケンスの追跡を行う機能の実装を行った．ログオンシーケンスの追跡は 4.2 章で述べる PsSetCreateProcessNotifyRoutine 関数を用いてプロセス生成イベントをフックし実装を行う．カーネルモードドライバで実装を行うため，ntoskrnel.exe のロード直後から全てのプロセス生成をフックできるため，ログオンシーケンスを観測することができる．ログオンシーケンスの追跡によるログオンセッション情報の収集では以下の手順で行う．

(1) winlogon.exe の生成を確認する．
(2) (1) で確認した winlogon.exe が親プロセスである userinit.exe の生成を確認する．
(3) (2) で確認した userinit.exe が親プロセスである explorer.exe の生成を確認する．
(4) (3) で確認した explorer.exe の SEP_LOGON_SESSION_REFERENCES 構造体からログオン情報を収集し，ログオンイベントの発生と断定する．

上記で示した一連の処理を観測した場合ログオンシーケンスと断定し，ログオンイベントの記録を行う．また，explorer.exe は管理者権限で実行されるため，ログオンユーザのログオン Id は SEP_LOGON_SESSION_REFERENCES の BuddyLogonId，管理者権限側のログオン Id は LogonId として収集を行う．

4.2 プロセス生成イベント

提案手法におけるプロセス生成イベントの記録実現するため，カーネルモードドライバを用いてプロセス生成のフックを行い，セッション情報と資格情報の収集機能の実装を行った．本章では，セッション情報と資格情報の収集機能を実装するために行ったプロセス生成のフック手法についての調査結果を用いてプロセス生成フックのコールバック関数内で実装したセッション情報収集機能，資格情報の収集機能について述べる．

4.2.1 プロセス生成のフック

プロセス生成・削除をカーネルモードでフックを行うためには，システムコールフックを行う方法とのプロセスマネージャルーチンの PsSetCreateProcessNotifyRoutine 系への登録を行う方法がある．システムコールフックでは，対象システムコールが呼び出された際の呼び出し先アドレスをコールバック関数に変更する手法である．この手法では，アドレステーブルの変更など環境依存の処理が必要となる．PsSetCreateProcessNotifyRoutine（Ex,Ex2）は Wnidows が提供するプロセス生成・終了のポストフック可能にする関数である．システムコールフックはプロセス生成などに限らず多くのシステムコールをフックできるといった利点がある．しかし，システムコールフックでは先述したように環境への依存度が高く，マイナーバージョ

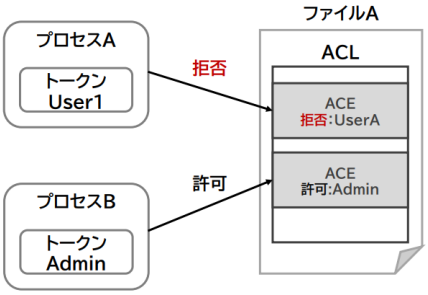

図 3 ファイル A に対するアクセス制御例

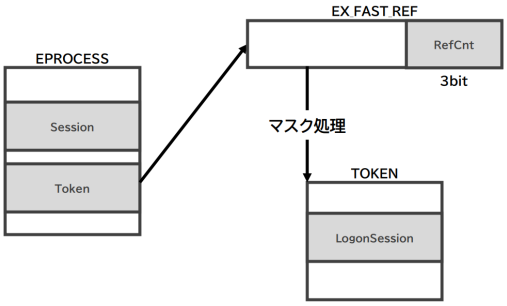

図 4 EPROCESS 構造体から資格情報を取得する手法

```
kd> dt nt!_EX_FAST_REF
   +0x000 Object           : Ptr64 Void
   +0x000 RefCnt           : Pos 0, 4 Bits
   +0x000 Value            : Uint8B
```

図 5 EX_FAST_REF 構造体

ンの変更などでも処理が失敗してしまう可能性がある．今回は，プロセス生成・終了といった特定の処理にのみ対応したいため，PsSetCreateProcessNotifyRoutine（Ex,Ex2）を利用する．

本手法で利用する PsSetCreateProcessNotifyRoutine 系の関数には 3 種類存在する．PsSetCreateProcessNotifyRoutineEx 関数, PsSetCreateProcessNotifyRoutineEx2 関数は, 登録する関数の引数として EPROCESS 構造体, プロセス ID, PS_CREATE_NOTIFY_INFO 構造体が渡される．本手法では，バージョンの差異を吸収するため，Windows Vista SP1 以降での利用を推奨される PsSetCreateProcessNotifyRoutineEx 関数と Windows 10 RS2 以降で利用を推奨される PsSetCreateProcessNotifyRoutineEx2 関数を併用する．

4.2.2 資格情報の取得

Windows 10 では，プロセスによるオブジェクトへのアクセスを，トークンと Access Contorol List（以下 ACL と記す）によって制御している．トークンは，プロセスの作成者や作成元セッションを管理しておりオブジェクトに対するアクセス権を証明する役割を持つ．プロセスの生成時に作成者であるユーザの持つトークンを継承することで割り当てられ，プロセスがファイルのようなオブジェクトに対してアクセスを要求した際にオブジェクト内で管理される ACL のエントリ Access Contorol Entry（以下 ACE と記す）と検証されアクセス可否を制御される．ファイル A に対する ACL によるアクセス制御例を図 3 に示す．

図 3 は，一般ユーザの User1 によって生成されたプロセス A と管理者権限を持つユーザ Admin によって生成されたプロセス B がファイル A に対するアクセス要求を出した際の ACL によるアクセス制御例である．プロセス A は一般ユーザ User1 によって生成されたためトークンには一般ユーザの権限と User1 によって生成されたことが管理されている．プロセス B は管理者権限を持つ Admin というユーザによって生成されたため，トークンには管理者権限と Admin によって生成されたことが管理されている．これらのプロセスがファイル A にアクセスした場合，ファイル A の ACL には UserA のアクセスを拒否する ACE と Admin のアクセスを許可する ACE が登録されているため，プロセス A のアクセスは拒否され，プロセス B のアクセスは許可される．そのため，資格情報を収集することでプロセスが管理者権限で実行されたかを特定できる．そこで，プロセスに継承された管理されるユーザの資格情報を取得する．

カーネル内部では，プロセスの持つ資格情報は TOKEN 構造体で管理される．そのため，フック時に取得可能な EPROCESS 構造体から資格情報を収集するためには，TOKEN 構造体へアクセスする必要がある．EPROCESS 構造体から TOKEN 構造体へアクセスする方法の概要を図 5 に示す．EPROCESS 構造体には Token メンバが存在し，Token メンバの EX_FAST_REF 構造体を用いることで TOKEN 構造体を参照できる．EX_FAST_REF 構造体は参照カウンタを含むポインタである．64bit 環境では下位 3bit が参照カウンタとなっており，TOKEN 構造体のアドレスは EX_FAST_REF の値に対して下位 3bit をマスクした結果となる．EX_FAST_REF 構造体の内容を図 5 に示す．そのため，EX_FAST_REF から参照カウンタをマスクし，TOKEN 構造体のアドレスを参照することで TOKEN 構造体の情報を取得することができる．

以上の方法でプロセス生成記録機能における資格情報の取得を実装した．

4.2.3 セッション情報の取得

Windows 10 では，4.1.2 節で述べたように仮想端末を分離するためターミナルセッションという概念が存在する．ターミナルセッションはセッション ID という識別子によって区別される．また，ユーザによるログオンを分離するため，ログオンセッションが存在し，ログオンセッションはログオン ID という識別子をによって区別される．カーネル内におけるプロセス構造体である ExecutiveProcess（以下 EPROCESS と記す）構造体では，プロセスの生成元セッション，ログオンセッションへのリンクを管理している．プロセスの生成元セッションを識別するために，以下の情

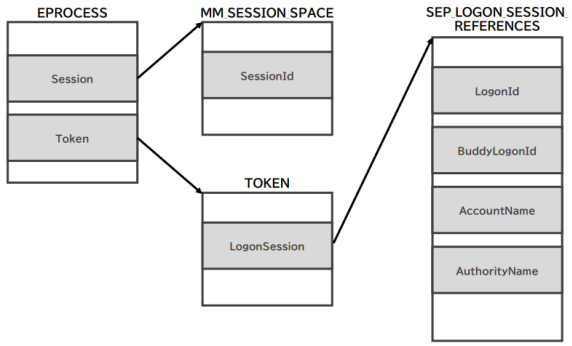

図 6 EPROCESS 構造体からセッション情報を取得する手法

報を EPROCESS 構造体から取得する．

- SessionId（セッション ID）
- LogonId（ログオン ID）
- BuddyLogonId
- AccountName
- AuthorityName

SessionId，LogonId はそれぞれ上記のセッション ID とログオン ID であり，プロセスの所属するセッション，ログオンセッションに対応する．BuddyLogonId は，管理者ユーザがログオンした際にログオンされる通常の権限のログオンセッション，管理者権限のログオンセッションのどちらかログオン ID でない側を管理する．AccountName，AuthorityName はそれぞれユーザがログオンセッションにログオンした際のアカウント名とドメイン名である．

上記の情報を EPROCESS 構造体から辿る方法を図 6 に示す．

図 6 では，SessionId を取得するためターミナルセッションを管理する MM_SESSION_SPACE を EPROCESS 構造体の Session メンバから参照できることを示している．また，LogonId，BuddyLogonId，AccountName，AuthorityName を取得するためログオンセッションを管理する SEP_LOGON_SESSION_REFERENCES 構造体を EPROCESS 構造体の Session メンバ，TOKEN 構造体の LogonSession メンバの順に辿ることで参照できることを示している．

4.3 ファイル変更イベント

4.3.1 ファイル変更のフック方法

Windows におけるファイル I/O では，I/O マネージャによって制御されミニフィルタードライバで処理が行われる．ミニフィルタドライバはファイル I/O における手続きを階層化しカーネルから分離することでファイルシステムドライバにおける冗長性の軽減，速度の向上を実現している．HDD などのハードウェアに対してファイルの I/O 要求が起きた際のカーネルでの処理内容を図 7 に示す．

I/O 要求が発生するとカーネル内部では，I/O マネージャによって要求の処理が行われ，フィルタマネージャに制御

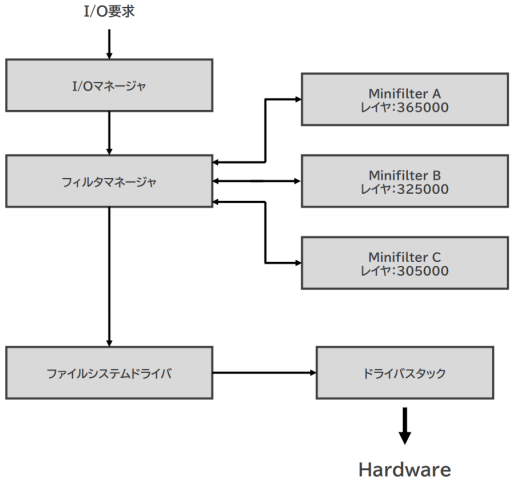

図 7 ファイル I/O におけるカーネル内の処理

が渡される．フィルターマネージャはフィルタドライバの階層に基づいて処理を制御する．例として，図 7 のカーネルでは，MinifilterA，MinifilterB，MinifilterC がロードされている．I/O 要求が発生すると階層の最も高い MinifilterA から処理が行われ，次に階層の高い MinifilterB，最後に MinifilterC が実行される．その後，下位層のアンチウイルスなどのフィルタードライバの処理に制御され，ファイルシステムドライバに受け渡される．ファイル変更イベントの記録手法では，ファイルの変更を観測し観測した情報を出力するため，配置可能な階層範囲としては Filter，FSFilter トップ，FSFilter アクティビティモニターが挙げられる．以上の調査から，提案手法におけるファイル変更履歴の記録ではファイルシステムに依存しないファイル I/O 処理のフックとしてミニフィルタドライバを用い手実装する．また，フィルタマネージャによって管理される他のフィルタードライバの影響を受けることを避けるため，登録可能な階層の中で最も上位である Filter オーダグループに階層を設定する．

4.3.2 プロセス情報の収集

フィルタドライバによってファイルの変更をフックした際に行うプロセス情報収集の実装について述べる．フィルタドライバでは，I/O 要求を行ったカレントプロセスのプロセス ID を取得する関数として GetCurrentProcess が用意されている．GetCurrentProcess 関数は引数なしの関数で，戻り値としてプロセスハンドルを返す．また，プロセスハンドルから EPROCESS 構造体を取得する PsLookupProcessByProcessId 関数では，引数にプロセスハンドルと受け取り用の EPROCESS 構造体を渡すことで，プロセスハンドルに対応する EPRROCESS 構造体を取得可能である．これらの関数を組み合わせて利用することによって，カレントプロセスの EPROCESS 構造体を取得する．ファイル変更イベントで収集すべき，作成元プロセス ID/名，作成元スレッド ID/名，作成元スレッドの GUI フラグを

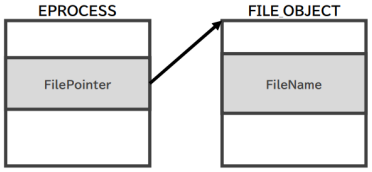

図 8 EPROCESS 構造体から実行ファイルのフルパスへの辿り方

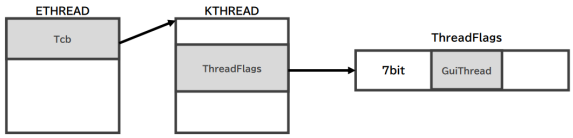

図 9 ETHREAD 構造体から ThreadFlags への辿り方

EPROCESS 構造体から実行ファイルのフルパスを取得する方法を図 8 に示す．

EPROCESS 構造体のメンバにファイル名を格納している ImageFileName がある．しかし，ファイル名でプロセスの実行ファイルの特定を行うのは不確実であり，実行ファイルのフルパスが求められる．そこで，実行ファイルのフルパスを取得するため図 8 で示す方法を用いてファイルオブジェクトの取得を行う．ファイルオブジェクトはファイルの情報を管理するカーネルオブジェクトであり，フルパスを FileName メンバで管理する．

以上の方法でファイル変更イベントにおけるプロセス情報の取得機能の実装を行った．プロセス ID や実行ファイルのフルパス以外の情報は，プロセス生成イベントで収集した情報と突合することによって対応可能である．

4.3.3 スレッド情報の収集

フィルタドライバによってファイルの変更をフックした際に行うスレッド情報収集の実装について述べる．フィルタドライバでは，I/O 要求を行ったプロセスのカレントスレッドの ETHREAD 構造体を返す PsGetCurrentThread 関数が用意されている．そこで，PsGetCurrentThread 関数を用いて，スレッドフラグ，スレッド ID の取得を行う．図 9 に ETHREAD 構造体から KTHREAD 構造体を辿り，ThreadFlags の GuiThread フラグを参照する方法を示す．

ETHREAD 構造体は EPROCESS 構造体と同様でカーネルモードにおけるスレッドの管理用構造体である．KTHREAD 構造体はスレッドにおける制御を管理する構造体である．KTHREAD 構造体では，ThreadFlags という long 型のフラグの集合があり，ThreadFlags のオフセット 7bit の位置で管理されているフラグが GUI スレッドのフラグとなっている．

また，スレッド ID は Cid から取得を行う．Cid は Windows におけるプロセス ID とスレッド ID を同時に管理している識別子で，スレッドの識別では Cid を利用することが多い．そのため，実装では ETHREAD 構造体のメンバの Cid を取得することでスレッド ID の収集の実現とする．

5. 評価

提案手法適応によってユーザがログオンしたセッションにおけるプロセスの挙動を特定可能であるかや，オーバヘッドによる影響について評価する．本章では，評価環境，機能評価，性能評価について述べる．

5.1 評価環境

提案手法では，カーネルモードドライバを用いて実装を行った．そのため，通常のアプリケーションと異なり致命的なエラーが起きる可能性がある．また，検証では Microsoft による署名のないカーネルモードドライバで動作検証を行うため，対象の OS をテストモードで起動する必要がある．そこで，仮想環境を用いた検証環境にて動作検証を行う．検証環境の構成を表 1 に示す．

表 1 評価環境

項目	内容
ホスト OS	Windows10 Version19042.746
仮想マシンモニタ	VMWare WorkStation
仮想 CPU コア数	2
割り当てメモリ	2GB
ゲスト OS	Windows10 Version19042.631

5.2 機能評価

本評価では，3 章で述べた，ログオン，プロセス生成，ファイル操作の各イベントにおいて対象とする情報が収集されているか評価する．

5.2.1 評価結果

図 10，11 では，notepad.exe における収集した各々の情報のメモリダンプと構造体に当てはめたものである．図 10 は，notepad.exe におけるプロセス情報を収集した結果である．コマンドラインとイメージファイルパスは，各々 PROCESS_INFO 構造体の CommandLine メンバと ImageFileName メンバのサイズで先頭アドレスから 0x1c のオフセットの位置にそれぞれ格納されていることを確認した．同様に，図 11 では，LOGON_INFO 構造体の UserName，AuthorityName メンバで示されるサイズで先頭アドレスから 0x20 のオフセットの位置にそれぞれ対応する情報が格納されていることを確認した．また，ファイル変更，ログオンについても同様に収集できていることを確認し，ログオン時に収集したセッションＩＤ，ログオンＩＤが notepad.exe で収集した情報と一致することを確認した．

本評価において，ユーザによって生成されたログオンセッションにて生成されたプロセスではセッション ID，ログオン ID などのセッション情報によって分離できることを示した．よって，提案手法によって収集した情報でユー

```
0: kd> dd 0xffffa50201277000
ffffa502`01277000  000000ac 00000000 000001c0 00000000
ffffa502`01277010  00000edc 00000000 00460046 ffffe185
ffffa502`01277020  00430022 005c003a 00690057 0064006e
ffffa502`01277030  0077006f 005c0073 00790073 00740073
ffffa502`01277040  006d0065 00320033 006e005c 0074006f
ffffa502`01277050  00700065 00640061 0065002e 00650078
ffffa502`01277060  00200022 005c0000 003f003f 0043005c
ffffa502`01277070  005c003a 00690057 0064006e 0077006f
0: kd> dt _PROCESS_INFO 0xffffa50201277000
KMDFDriver1!_PROCESS_INFO
   +0x000 Size             : 0xac
   +0x008 Pid              : 0x00000000`000001c0 Void
   +0x010 PPid             : 0x00000000`00000edc Void
   +0x018 CommandLine      : 0x46
   +0x01a ImageFileName    : 0x46
```

図 10 notepad.exe におけるプロセス情報

```
0: kd> dd 0xffffa50201278000
ffffa502`01278000  0000004a 00000000 00091715 00000000
ffffa502`01278010  000916e7 00000000 00091413 0020000a
ffffa502`01278020  00650074 00740073 00440000 00530045
ffffa502`01278030  0054004b 0050004f 0036002d 00430046
ffffa502`01278040  00470038 00300045 00000000 00000000
ffffa502`01278050  00000000 00000000 00000000 00000000
ffffa502`01278060  00000000 00000000 00000000 00000000
ffffa502`01278070  00000000 00000000 00000000 00000000
0: kd> dt _LOGON_INFO 0xffffa50201278000
KMDFDriver1!_LOGON_INFO
   +0x000 Size             : 0x4a
   +0x008 LogonId          : _LUID
   +0x010 BuddyLogonId     : _LUID
   +0x018 SessionId        : 0x91413
   +0x01c UserName         : 0xa
   +0x01e AuthorityName    : 0x20
```

図 11 notepad.exe におけるセッション情報

ザによってログオンされたセッション内のプロセスの挙動を識別することができる．

5.3 性能評価

本評価では，提案手法適応時に掛かるオーバヘッドを計測するため，本手法を適用した仮想計算機において 1000 回のイベントを観測した際の平均値を計測した．

5.3.1 評価方法

提案手法では，カーネルオブジェクト内で管理される情報の収集を行う．そのため，ms オーダーでの計測では正確な計測が行われない可能性がある．そこで，カーネル内で µs オーダーでの高精度のタイムスタンプを取得可能な KeQueryPerformanceCounter 関数を用いる．KeQueryPerformanceCounter 関数は，クロック周波数が一定なハードウェアタイマを利用するため，クロック周波数で除算することで，システム稼働時からの時間を計測することができる．そこで，この方法を用いて提案手法の適用前後の時間を計測し，差分をとることでオーバヘッドの計測を行う．

5.4 評価結果

1000 イベントを計測した際のオーバヘッドの平均値は，約 371.3µs となった．提案手法では，ユーザ操作時のカーネルオブジェクトの情報と正確なタイムスタンプを必要とする．タイムスタンプで利用される Unix Time では環境によって µs オーダでの記録が可能だが，提案手法のオーバヘッドはユーザ操作に著しく影響するオーバヘッドではない．

6. まとめ

本論文では，ユーザによって操作されたプロセスの挙動を特定可能なカーネルオブジェクト内の情報を記録する手法についてについて述べた．ログオンイベント，プロセス生成イベント，ファイル変更イベントについて対象の情報を収集し，検証にてユーザによるイベントを特定できることを確認した．ログオンイベントでは，ログオンフックで収集ができないユーザ情報について，ログオンシェルから収集することでログオンイベントにおけるユーザ情報の収集を可能とした．プロセス生成イベントでは，プロセスの所属するセッション情報や生成したユーザの資格情報を収集することで上記のセッション情報との対応付け可能とした．また，ファイル変更イベントでは，当該ファイルの変更を行ったプロセスとスレッドの情報を収集することでユーザのログオンしているセッション内でのファイル編集の特定を補助する情報を残すことを可能とした．しかし，ユーザのデスクトップ画面を録画する攻撃事例 [5][6] も確認されており，GUI アプリケーションの模擬を行う必要もある．そこで，今後 GUI アプリケーションにおける GUI コンポーネントの操作記録と組み合わせることで対応することを検討する．

参考文献

[1] 警察庁：令和元年におけるサイバー空間をめぐる脅威の情勢等について (2020).
[2] 津田侑，遠峰隆史，金谷延幸，牧田大佑，丑丸逸人，神宮真人，高野祐輝，安田真悟，三浦良介，太田悟史，宮地利幸，神薗雅紀，衛藤将史，井上大介，中尾康二：サイバー攻撃誘引基盤 STARDUST，コンピュータセキュリティシンポジウム 2017 論文集，Vol. 2017, No. 2 (2017).
[3] 竹久達也，牧田大佑，神宮真人，丑丸逸人，福森大喜， 津田侑，遠峰隆史，井上大介：デバイスドライバを用いたプロセス挙動保全ツールの提案，コンピュータセキュリティシンポジウム 2017 論文集，Vol. 2017, No. 2 (2017).
[4] Microsoft: Sysmon - Windows Sysinternals — Microsoft Docs. 入手先⟨https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon⟩（参照 2020-12-10）.
[5] Report, T.: CARBANAK Targeted Attack Campaign Hits Banks and Financial Institutions（オンライン）. 入手先⟨https://www.trendmicro.com/vinfo/jp/threat-encyclopedia/web-attack/3142/carbanak-targeted-attack-campaign-hits-banks-and-financial-institutions⟩（参照 2020-12-10）.
[6] Repor, K.: CARBANK APT（オンライン）. 入手先⟨https://media.kaspersky.com/jp/pdf/pr/Kaspersky$_W$P−Carbanak − PR − 1011.pdf⟩ΓΓ2020 − 12 − 10Γ.