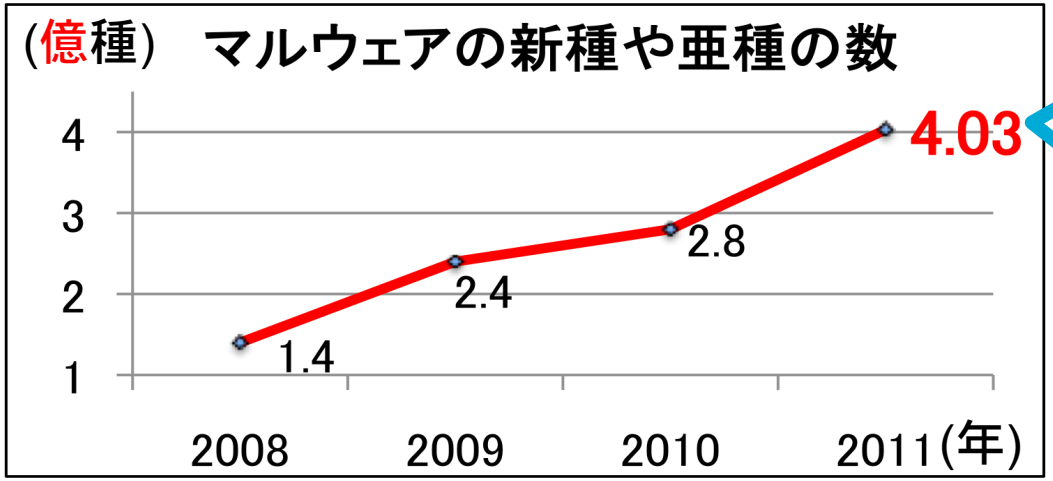
BitVisorへのシステムコールトレース機能追加によるマルウェアアナライザの実現

大月 勇人, 瀧本 栄二, 毛利 公一

立命館大学

背景



マルウェアが急速に増加！

2011年には**4億300万種**の新種や亜種が出現！

(Symantec のデータより†)

- 短時間で解析し，マルウェアの意図や概略を把握したい
  - マルウェアを実行し，挙動を観測することで解析する動的解析が有効
- しかし，マルウェアの巧妙化により，観測自体が困難となっている
  - アンチデバッグ:
    観測ツールを検知し，観測・解析を妨害する
  - コードインジェクション:
    一般のプロセスに感染し，「悪意あるスレッド」を潜ませる

† http://www.symantec.com/ja/jp/about/news/release/article.jsp?prid=20100428_02,
http://www.symantec.com/ja/jp/about/news/release/article.jsp?prid=20110412_01,
http://www.symantec.com/ja/jp/about/news/release/article.jsp?prid=20120501_01

マルウェア観測システムの条件

- マルウェアに検知されない観測システムの実現
  - マルウェアよりも高い権限で動作
  - マルウェア動作環境への影響を抑制

  仮想計算機モニタ(VMM)として実現
- VMM を検出されないために
  - ハードウェア構成を固定しない

    準パススルー型 VMM BitVisor をベースにする
  - ゲスト OS と通信せずに内部の情報を得る

    ゲストOSのメモリの内容を解釈し，情報を取得

BitVisor の利点

- 実環境に近い
  - ゲスト OS は実マシンのハードウェアを認識する
    - マルウェアに検出されにくい
  - オーバヘッドが小さい
    - 1体辺りの解析時間の短縮,
      時間計測を用いたアンチデバッグの回避などが見込める
- 脆弱性が潜在する可能性が低い
  - マルウェアに乗っとられる可能性が低くなる
- VM が1つのみ
  - 解析中に他の VM の影響を受けない

マルウェア観測手法

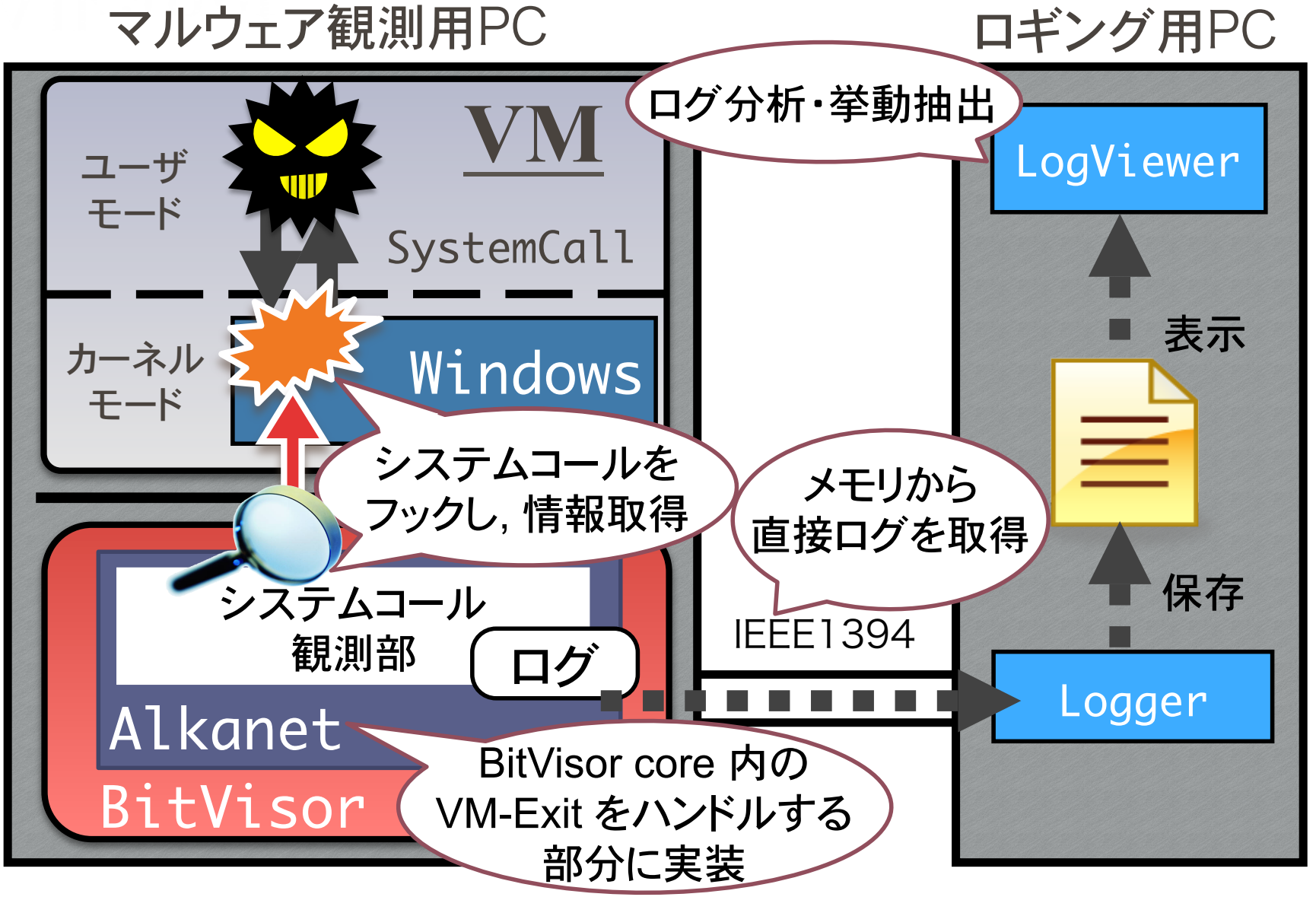
- マルウェアの意図を理解しやすい情報を提供
  - 粒度の観点から命令単位よりもAPI単位の観測が有効
  - 悪意あるスレッドがシステムに影響を与えるには
    システムコールが必要

システムコールトレースを基にした挙動解析

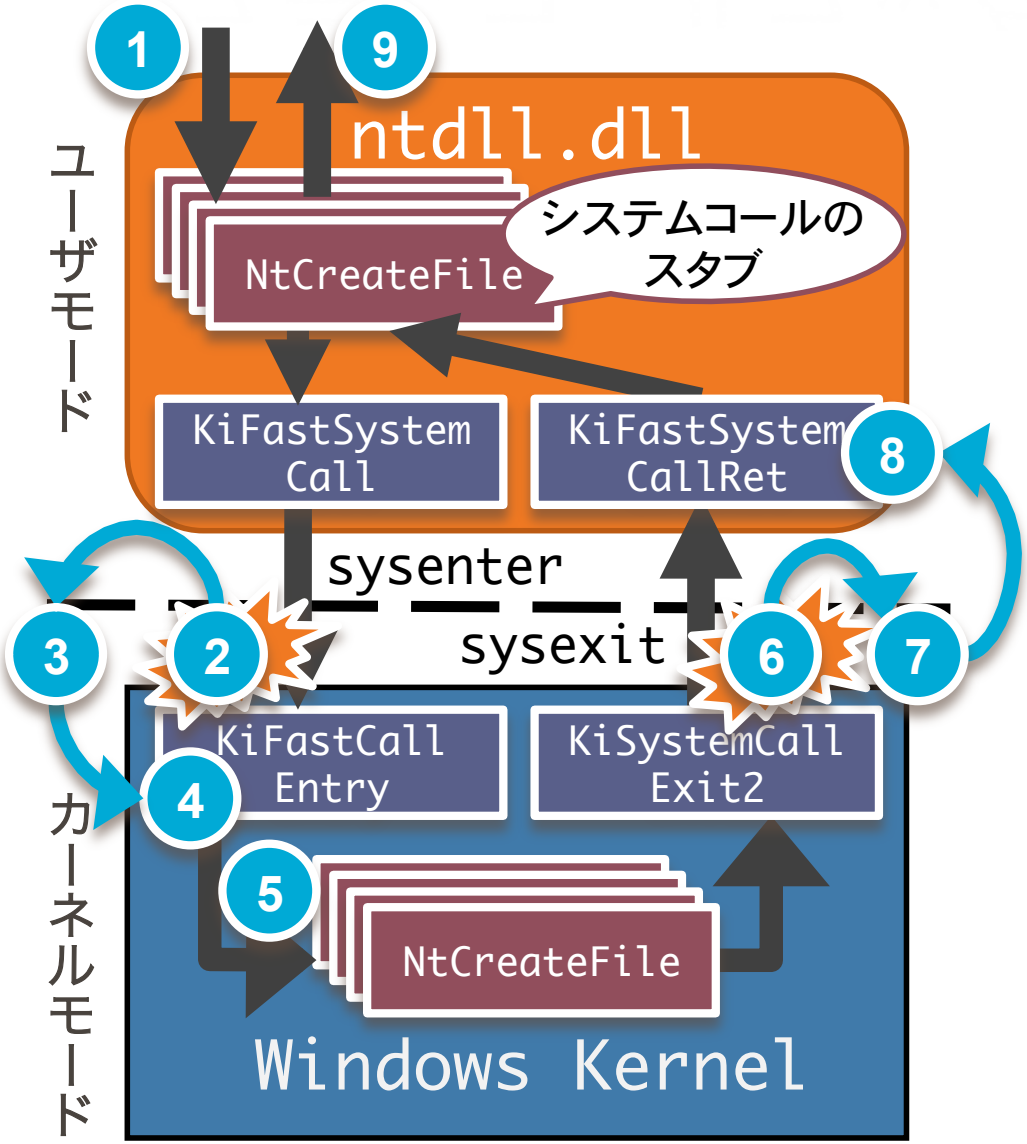
- 悪意あるスレッドを追跡し，挙動を観測
  - コードインジェクションを構成する挙動を観測
    - 別プロセスへのメモリ書換え，DLL 挿入，スレッド作成など
  - システムコールの発行元をスレッドレベルで区別

作成されたスレッドの情報と発行元の情報から
悪意あるスレッドを追跡

Alkanet

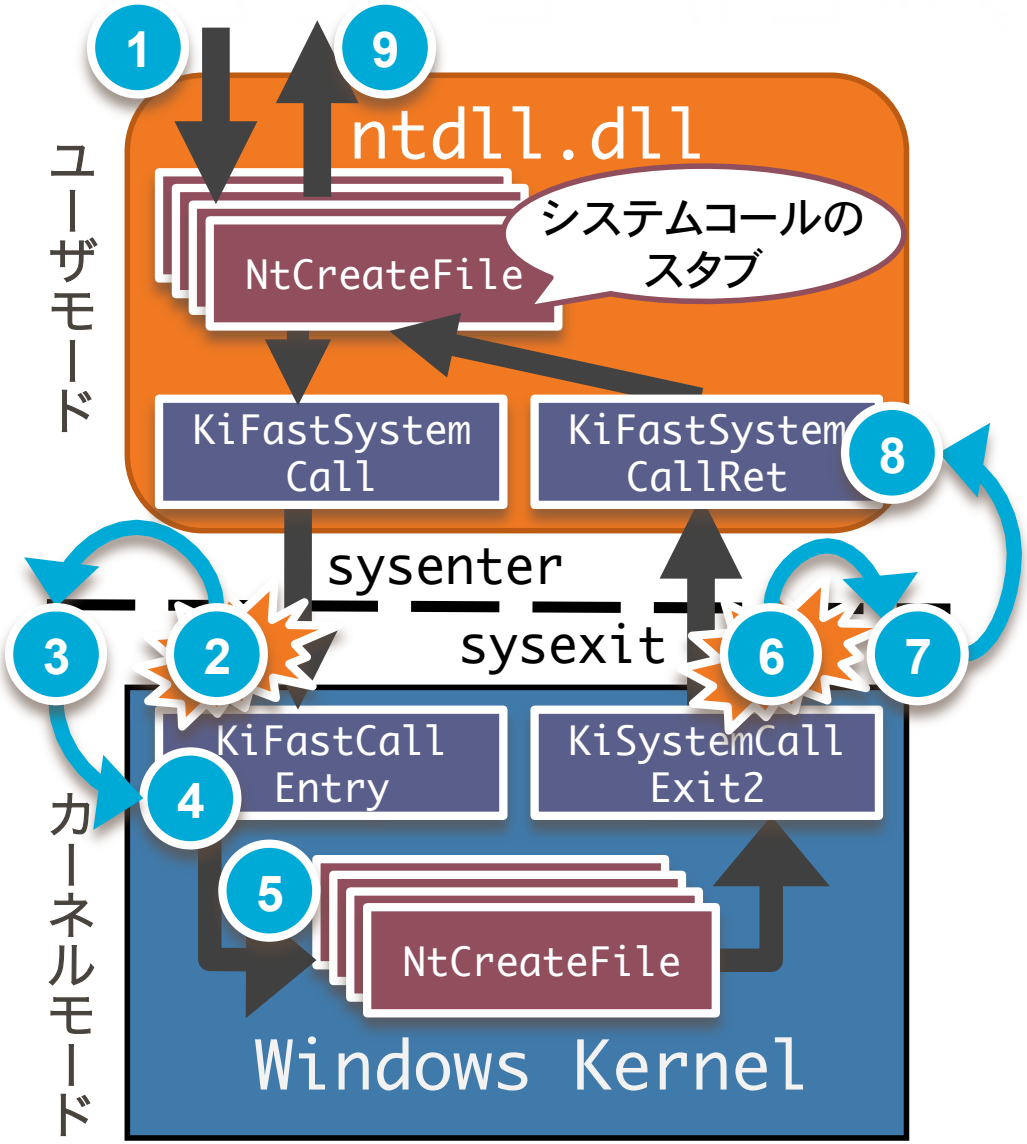

マルウェア観測用PC
ロギング用PC
VM
ユーザ
モード
SystemCall
カーネル
モード
Windows
ログ分析・挙動抽出
LogViewer
表示
システムコールを
フックし，情報取得
メモリから
直接ログを取得
システムコール
観測部
ログ
保存
IEEE1394
Logger
Alkanet
BitVisor
BitVisor core 内の
VM-Exit をハンドルする
部分に実装

システムコールフックの流れ (1/2)



1. マルウェアが
   システムコールを発行する
2. ブレイクポイントにより,
   VM-Exit を発生させ
   Alkanet に制御を移す
   - HWBP を使用
   - BitVisor にDR隠蔽機能を追加
3. 必要な情報の取得する
   - システムコール発行元
     プロセスとスレッドの情報
     - プロセスID, スレッドID,
       プロセスの名前
   - システムコール番号
   - システムコールの引数
   - 固有のデータ構造に
     対する補足情報
4. Windows に制御を戻す

システムコールフックの流れ (2/2)

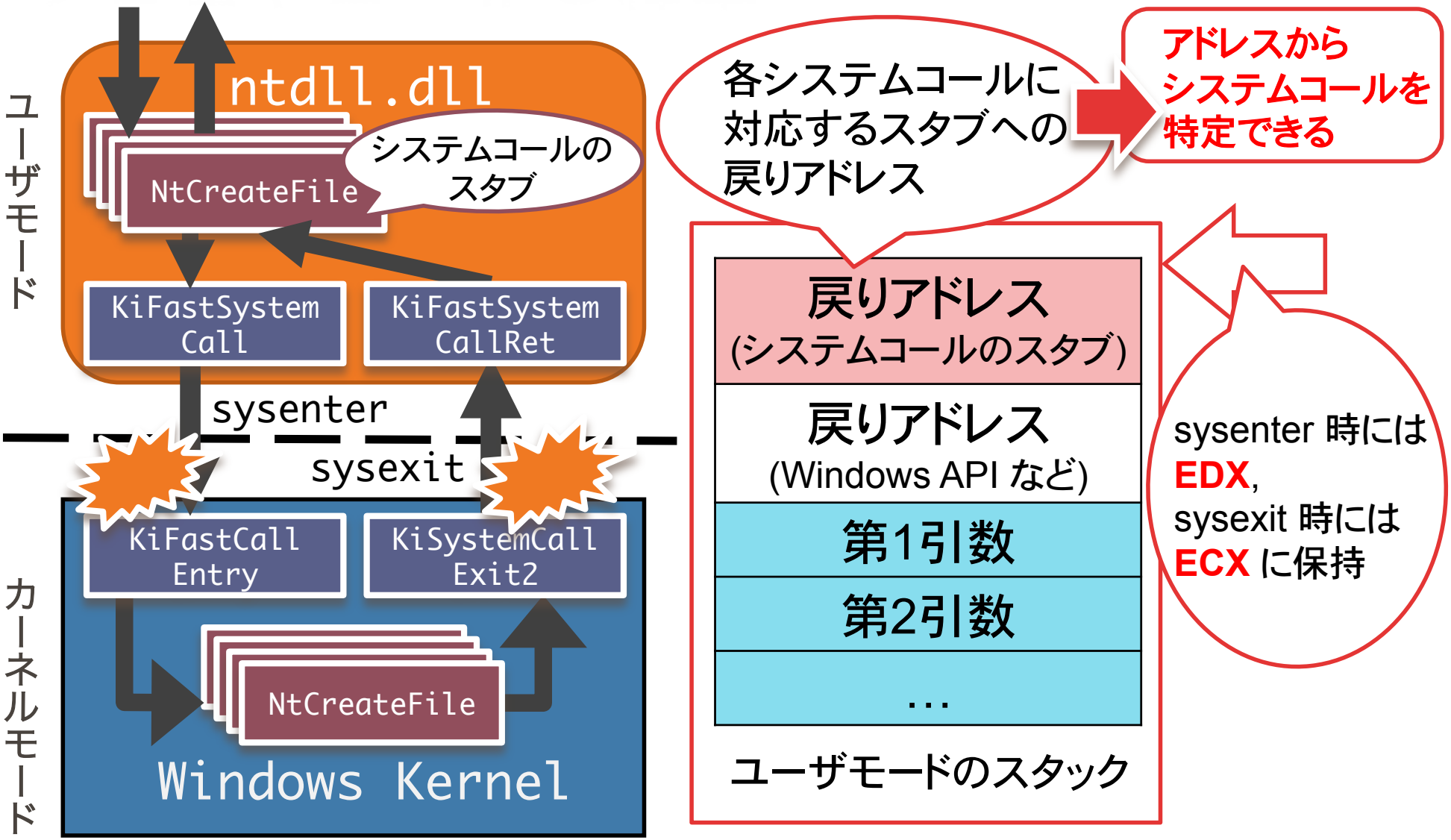


5. システムコールが実行される
6. ブレイクポイントにより Alkanet に制御を移す
7. システムコールの結果を取得する
   - 戻り値や生成された**オブジェクトの情報**など

スレッド作成の挙動の場合，作成されたスレッドの情報

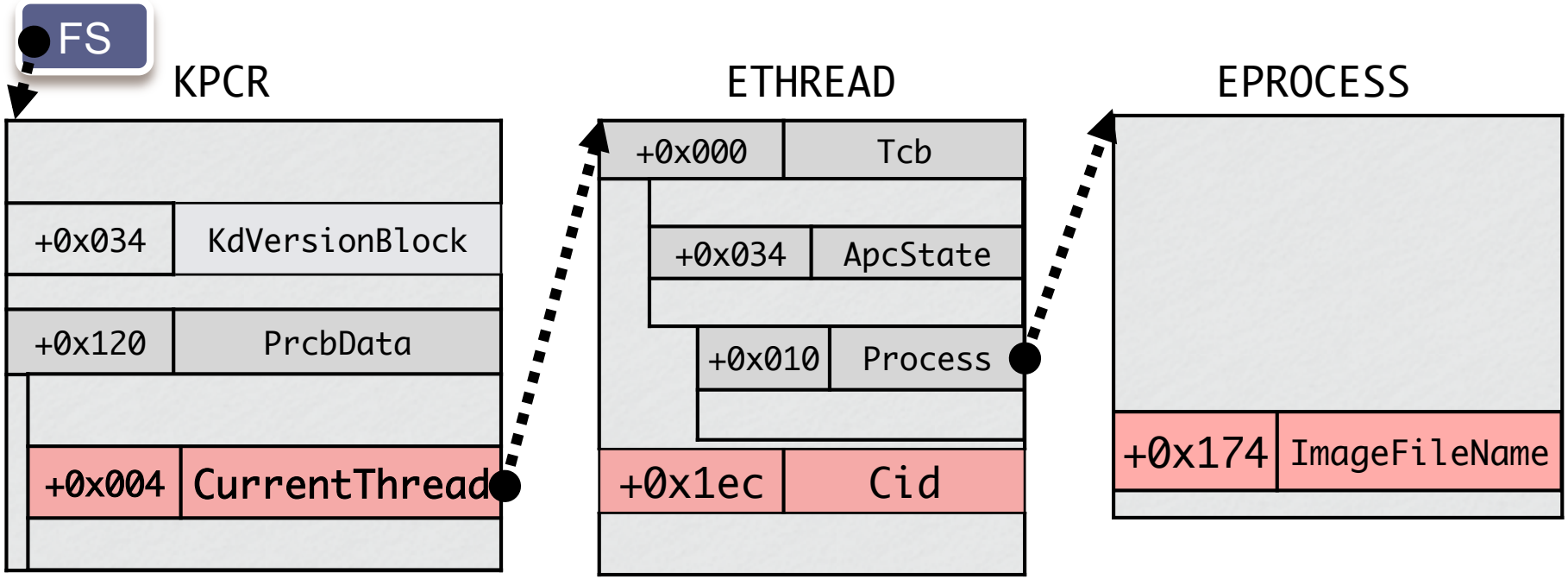
8. Windows に制御を戻す
9. マルウェアに制御が戻る

システムコールの特定


ユーザモード
ntdll.dll
NtCreateFile
システムコールのスタブ
KiFastSystemCall
KiFastSystemCallRet
sysenter
sysexit
KiFastCallEntry
KiSystemCallExit2
NtCreateFile
Windows Kernel
カーネルモード
各システムコールに対応するスタブへの戻りアドレス
アドレスからシステムコールを特定できる
戻りアドレス
(システムコールのスタブ)
戻りアドレス
(Windows API など)
第1引数
第2引数
...
ユーザモードのスタック
sysenter 時には EDX,
sysexit 時には ECX に保持

システムコール発行元の識別方法

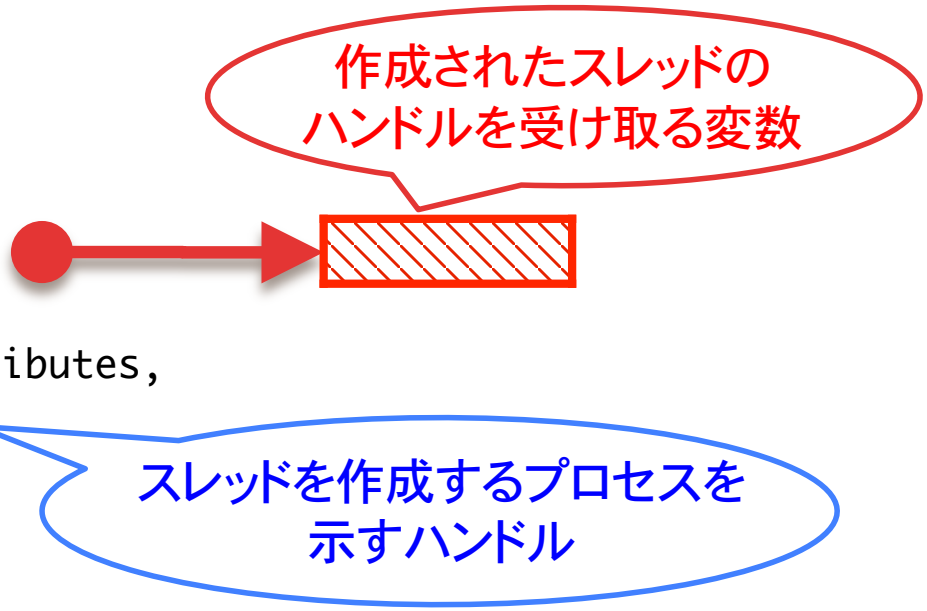
- PCR: 各プロセッサの状態を保持するデータ構造
  - カーネルや HAL, ドライバから利用される
  - カーネルモードでは FS セグメントにマップされている
- Alkanet では, このデータ構造を解釈・利用する
  - 現在実行中のスレッドオブジェクトへのポインタ
  - カーネルデバッガ用の構造体へのポインタ .....など



システムコールの引数と戻り値

- 引数
  - IN: カーネル側に渡す値
  - OUT: カーネル側が返す結果を受け取る変数へのポインタ
- 戻り値(NTSTATUS):
  システムコールの成否, 失敗の理由などを示す値

```
NTSYSAPI
NTSTATUS
NTAPI
NtCreateThread(
    OUT PHANDLE ThreadHandle,
    IN ACCESS_MASK DesiredAccess,
    IN POBJECT_ATTRIBUTES ObjectAttributes,
    IN HANDLE ProcessHandle,
    OUT PCLIENT_ID ClientId,
    IN PCONTEXT ThreadContext,
    IN PUSER_STACK UserStack,
    IN BOOLEAN CreateSuspended
    );
```



Alkanet が監視するシステムコール

挙動	システムコールの例
ファイル	NtCreateFile, NtReadFile, NtWriteFile, …
レジストリ	NtQueryValueKey,NtSetValueKey,…
仮想メモリ	NtWriteVirtualMemory, NtProtectVirtualMemory, …
ファイルマッピング	NtCreateSection, NtOpenSection, NtMapViewOfSection, …
プロセス	NtCreateProcessEx, NtTerminateProcess, …
スレッド	NtCreateThread,NtTerminateThread, NtSetContextThread, …
ネットワーク	NtDeviceIoControlFile, …
ドライバ	NtLoadDriver, NtUnloadDriver

Alkanet が監視するシステムコール

挙動	システムコールの例
ファイル	NtCreateFile, NtReadFile, NtWriteFile, …
レジストリ	NtQueryValueKey,NtSetValueKey,…
仮想メモリ	NtWriteVirtualMemory, NtProtectVirtualMemory, …
ファイルマッピング	NtCreateSection, NtOpenSection, NtMapViewOfSection, …
プロセス	NtCreateProcessEx, NtTerminateProcess, …
スレッド	NtCreateThread,NtTerminateThread, NtSetContextThread, …
ネットワーク	NtDeviceIoControlFile, …
ドライバ	NtLoadDriver, NtUnloadDriver

コードインジェクションに用いられるシステムコールへ対応

メモリ書き込み/スレッド挿入の挙動

- 以下は “Polipos” というマルウェアを解析したときのログ
- 2分程度のトレースで 24,000 エントリのログを取得

No. / Time	6337 / 689820579
Type	sysenter
SNo.	115 (NtWriteVirtualMemory)
Invoker	bc.304, Polipos.exe
Note	PID: b0, ProcessName: explorer.exe

No. / Time	6339 / 689820849
Type	sysenter
SNo.	35 (NtCreateThread)
Invoker	bc.304, Polipos.exe
Note	PID: b0, ProcessName: explorer.exe

No. / Time	6338 / 689820647
Type	sysexit
Ret	0 (STATUS_SUCCESS)
SNo.	115 (NtWriteVirtualMemory)
Invoker	bc.304, Polipos.exe
Note	PID: b0, ProcessName: explorer.exe

No. / Time	6340 / 689820959
Type	sysexit
Ret	0 (STATUS_SUCCESS)
SNo.	35 (NtCreateThread)
Invoker	bc.304, Polipos.exe
Note	Cid: b0.1e8, ProcessName: explorer.exe

コードインジェクションの挙動の流れ

1. スレッド作成

```
108.118 Polipos.exe NtCreateThread                370.11c winlogon.exe => STATUS_SUCCESS
```

2. スレッドのコンテキスト取得

```
108.118 Polipos.exe NtGetContextThread            370.11c winlogon.exe => STATUS_SUCCESS
```

3. メモリ確保 & 権限設定

```
108.118 Polipos.exe NtAllocateVirtualMemory       370 winlogon.exe, BaseAddress: 0xd90000,
                                                  Protect: 0x40 (PAGE_EXECUTE_READWRITE), …
                                                  => STATUS_SUCCESS
108.118 Polipos.exe NtProtectVirtualMemory        370 winlogon.exe, BaseAddress: 0xd90000,
                                                  NewProtect: 0x40 (PAGE_EXECUTE_READWRITE),…
                                                    => STATUS_SUCCESS
```

4. メモリ書込み

```
108.118 Polipos.exe NtWriteVirtualMemory          370 winlogon.exe => STATUS_SUCCESS
```

5. スレッドのコンテキスト設定

```
108.118 Polipos.exe NtSetContextThread            370.11c winlogon.exe => STATUS_SUCCESS
```

6. スレッドの実行開始

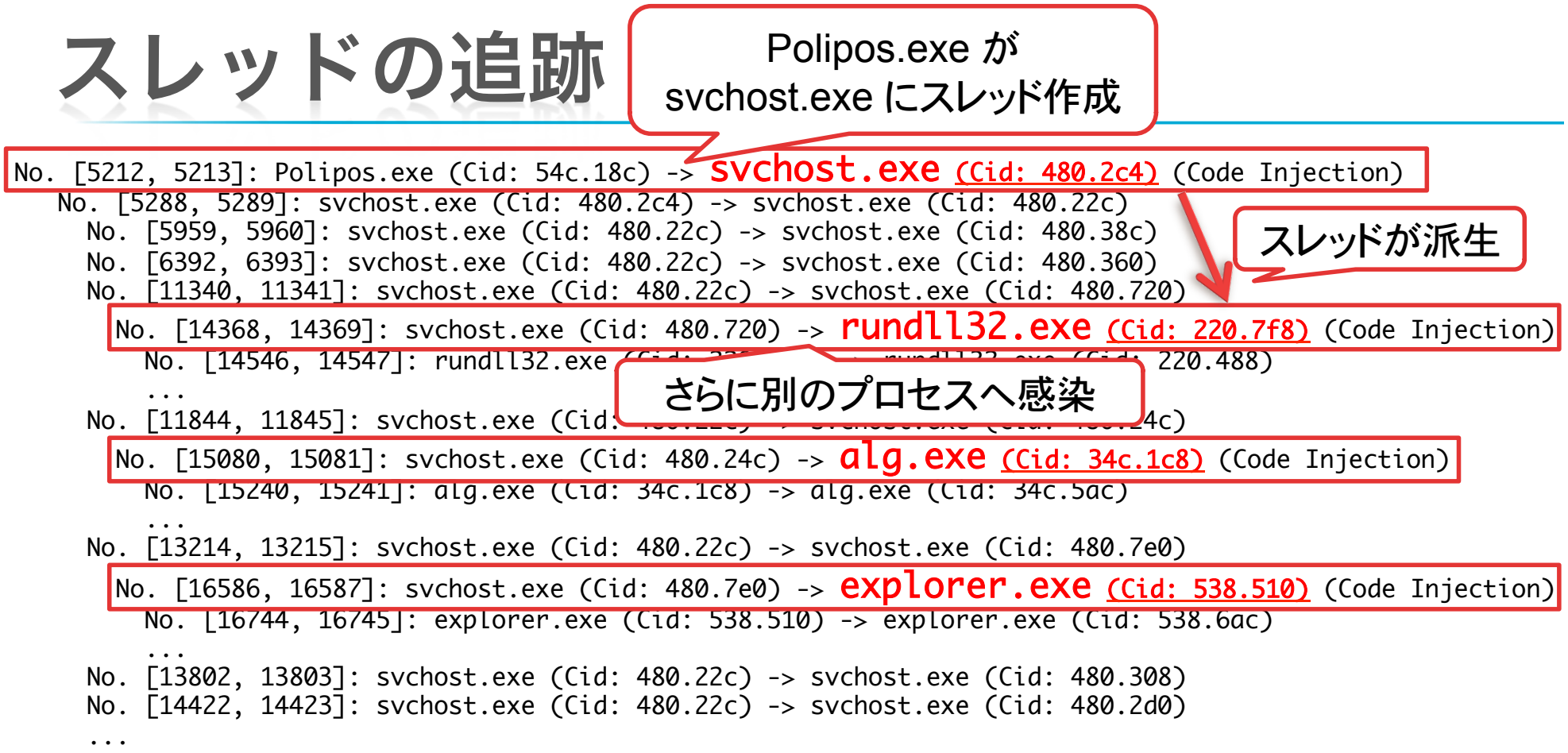
```
108.118 Polipos.exe NtResumeThread                370.11c winlogon.exe => STATUS_SUCCESS
```

スレッドの追跡

```
No. [5212, 5213]: Polipos.exe (Cid: 54c.18c) -> svchost.exe (Cid: 480.2c4) (Code Injection)
   No. [5288, 5289]: svchost.exe (Cid: 480.2c4) -> svchost.exe (Cid: 480.22c)
     No. [5959, 5960]: svchost.exe (Cid: 480.22c) -> svchost.exe (Cid: 480.38c)
     No. [6392, 6393]: svchost.exe (Cid: 480.22c) -> svchost.exe (Cid: 480.360)
     No. [11340, 11341]: svchost.exe (Cid: 480.22c) -> svchost.exe (Cid: 480.720)
       No. [14368, 14369]: svchost.exe (Cid: 480.720) -> rundll32.exe (Cid: 220.7f8) (Code Injection)
         No. [14546, 14547]: rundll32.exe (Cid: 220.7f8) -> rundll32.exe (Cid: 220.488)
         ...
     No. [11844, 11845]: svchost.exe (Cid: 480.22c) -> svchost.exe (Cid: 480.24c)
       No. [15080, 15081]: svchost.exe (Cid: 480.24c) -> alg.exe (Cid: 34c.1c8) (Code Injection)
         No. [15240, 15241]: alg.exe (Cid: 34c.1c8) -> alg.exe (Cid: 34c.5ac)
         ...
     No. [13214, 13215]: svchost.exe (Cid: 480.22c) -> svchost.exe (Cid: 480.7e0)
       No. [16586, 16587]: svchost.exe (Cid: 480.7e0) -> explorer.exe (Cid: 538.510) (Code Injection)
         No. [16744, 16745]: explorer.exe (Cid: 538.510) -> explorer.exe (Cid: 538.6ac)
         ...
     No. [13802, 13803]: svchost.exe (Cid: 480.22c) -> svchost.exe (Cid: 480.308)
     No. [14422, 14423]: svchost.exe (Cid: 480.22c) -> svchost.exe (Cid: 480.2d0)
     ...
```

- システムコールのログから, 別プロセスへ挿入された スレッドを追跡可能であることを確認
- ログ分析ツールを利用することで, マルウェアの挙動をさらに理解しやすくできる

スレッドの追跡

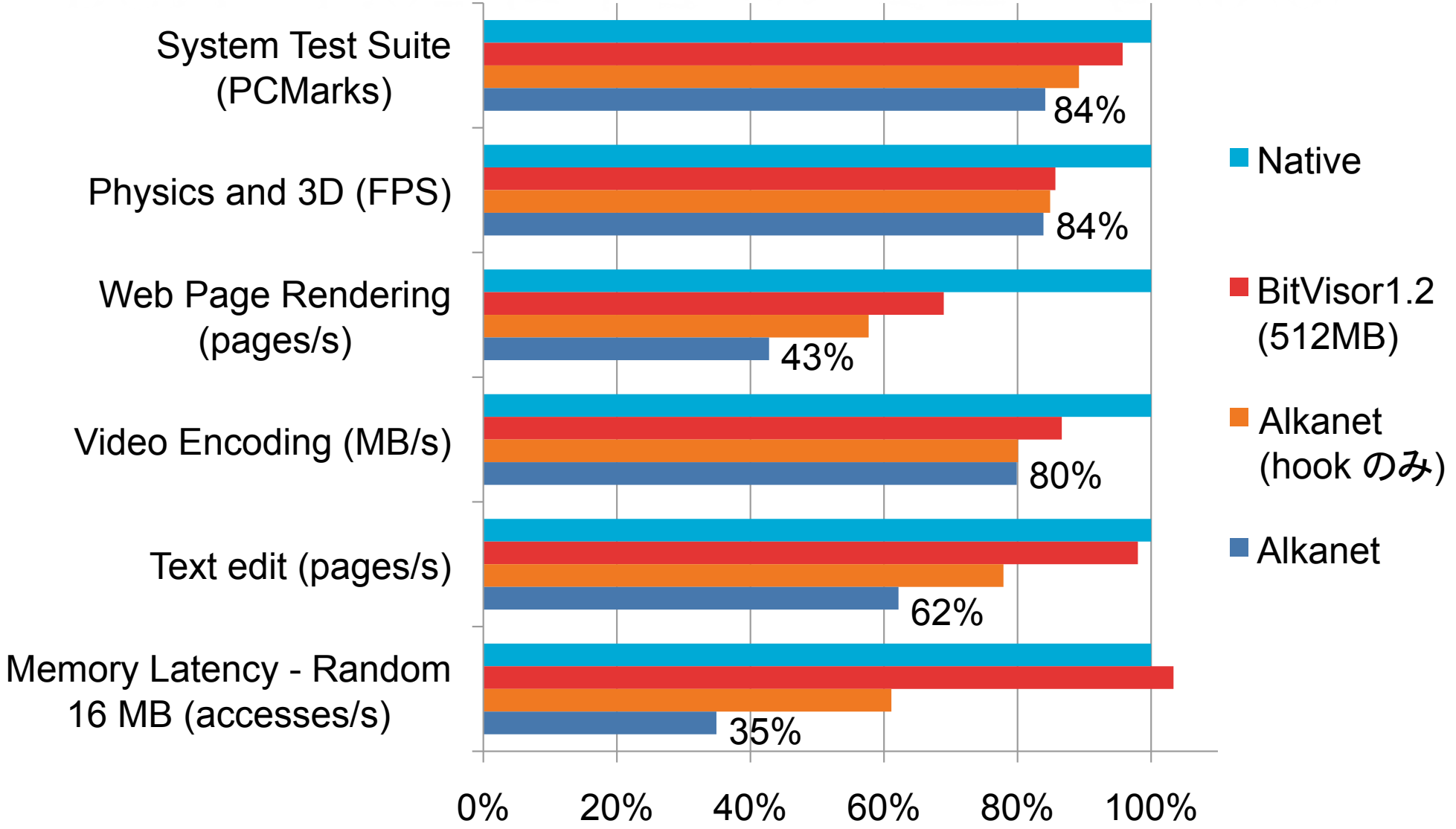


- システムコールのログから, 別プロセスへ挿入された
  スレッドを追跡可能であることを確認
- ログ分析ツールを利用することで,
  マルウェアの挙動をさらに理解しやすくできる

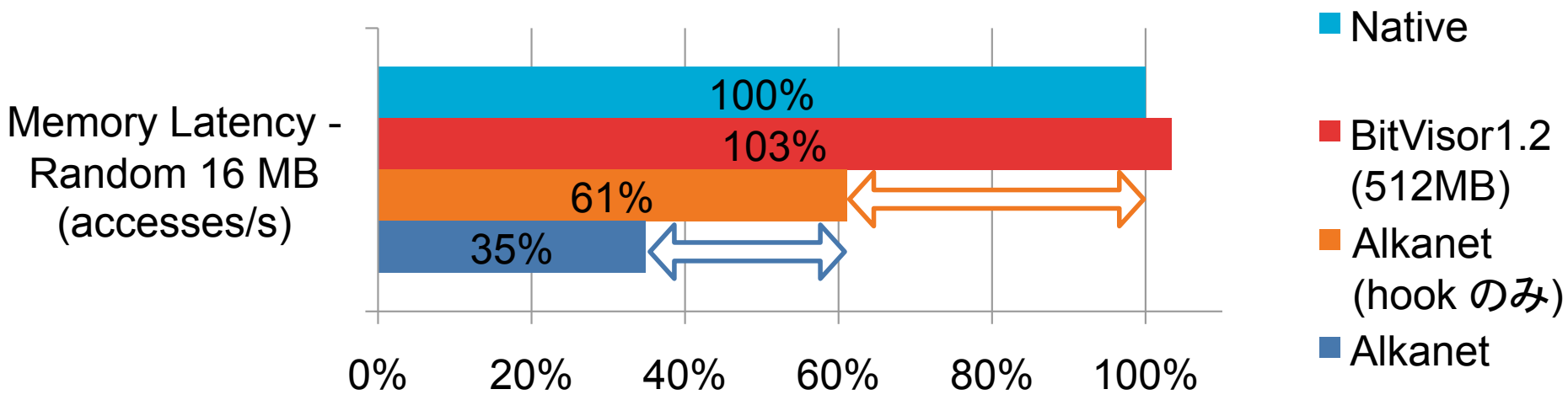
PCMark'05 によるベンチマーク (1/2)

Benchmark	Native	BitVisor1.2 (512MB)	Alkanet (hook のみ)	Alkanet
System Test Suite (PCMarks)	5324	5097	4749	4480
HDD - XP Startup (MB/s)	7.67	7.66	7.63	7.68
Physics and 3D (FPS)	150.29	128.75	127.57	126.06
Transparent Windows (windows/s)	301.66	307.17	304.87	299.63
3D - Pixel Shader (FPS)	57.33	57.04	57.05	56.86
Web Page Rendering (pages/s)	4.09	2.82	2.36	1.75
File Decryption (MB/s)	66.51	66.27	66.33	66.4
Graphics Memory - 64 lines (FPS)	484.61	484.42	485.71	487.57
HDD - General Usage (MB/s)	4.91	4.9	4.85	4.89
Audio Compression (MB/s)	3060.92	3036.65	2979.92	3012.35
Video Encoding (MB/s)	430.65	373.05	344.86	344
Text edit (pages/s)	147.53	144.61	114.91	91.72
Image Decompression (pixels/s)	34.55	34.14	34.24	34.24
File Compression (MB/s)	9.14	9.24	9.03	10.4
File Encryption (MB/s)	63.28	63.54	62.87	62.97
HDD - Virus Scan (MB/s)	69.94	67.26	64.18	64.96
Memory Latency - Random 16 MB (accesses/s)	11.03	11.4	6.74	3.85

PCMark'05 によるベンチマーク (2/2)



考察: Memory Latency 低下の理由



- ⇔: メモリ確保・操作のシステムコールのフック
- ⇔
  - Alkanet による Windows のメモリ領域の読み取り
  - ログのバッファリング
  - IEEE1394 の DMA によるログ取得

まとめ

- Alkanet
  - VMM を用いてアンチデバッグを回避する
    - BitVisor を用いることで実環境に近いマルウェア実行環境, 速度が実現可能
  - スレッド単位でマルウェアを追跡し, システムコールをトレースする
  - ログを元にマルウェアの特徴的な挙動を抽出するツール群も作成
  - 別プロセス内に作成されたスレッドも追跡可能であることを確認
- PCMark’05 によるベンチマーク
  - 総合では16%程度のオーバヘッド, 特に Memory Latency が65%低下
- 今後の課題
  - 評価の続き
    - オーバヘッドの評価
    - アンチデバッグに対する評価
  - ネットワークを用いた挙動を観測する機能の強化
    - 別のコンピュータに攻撃させないように配慮する必要がある
    - 既存のハニーポットとの連携
  - システムコールトレースログを元にした既存手法に Alkanet のログが利用できるか評価を行う
    - 異常検知やマルウェアのクラスタリングなど