NTT

NTT Secure Platform Laboratories
NTT セキュアプラットフォーム研究所

テイント伝搬に基づく解析対象コードの追跡方法

NTTセキュアプラットフォーム研究所
◎川古谷裕平、塩治榮太朗、岩村誠、針生剛男

目次

- 背景
- 問題定義
- 提案手法
- 実装
- 実験
- 関連研究
- 考察
- まとめ

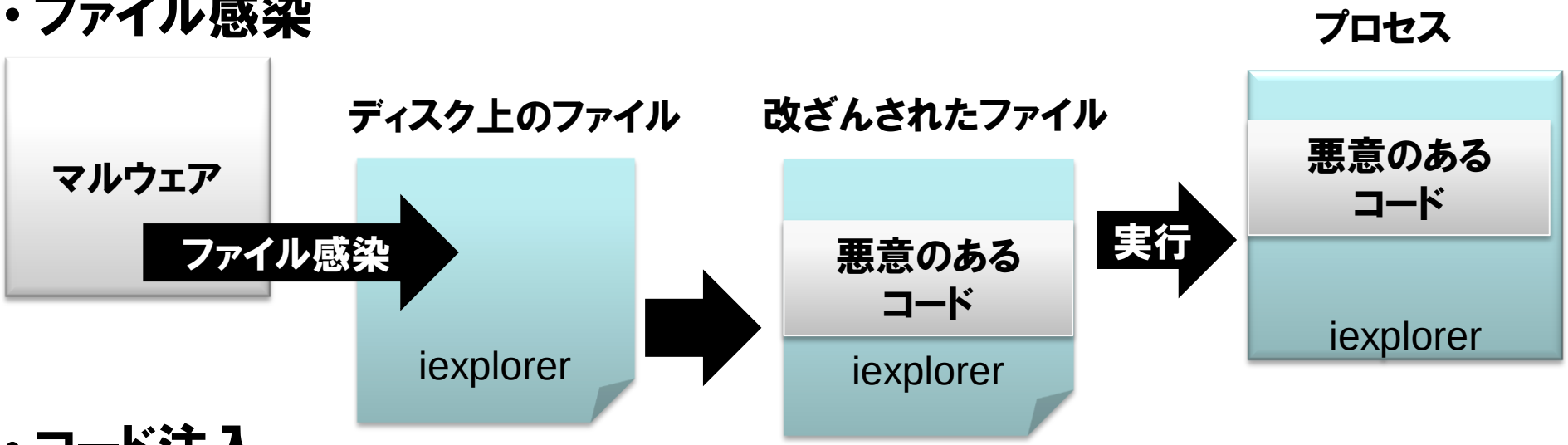
背景

- マルウェアの動的解析が様々なところで利用されている
- 解析対象コードとそれ以外を区別する必要がある
  - OSの中には多数のコードが動作している
- プロセスID(PID)、スレッドID(TID)による方法が一般的
  - TTAnalyze(Anubis), TEMU, Ether etc…

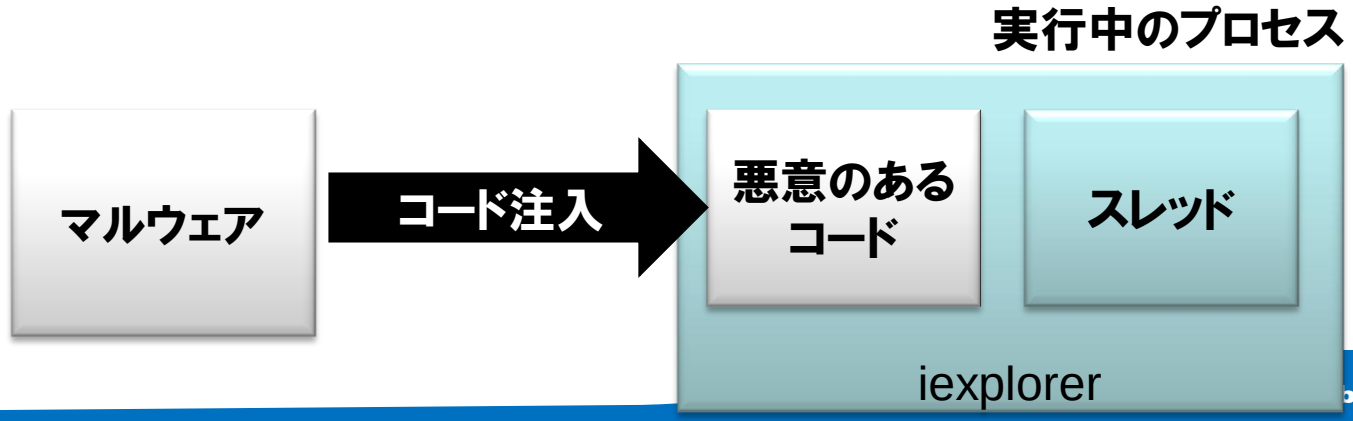
解析妨害機能

- マルウェアの解析妨害機能
  - 実行環境検知(デバッガ、仮想マシン、etc…)
  - 解析対象回避(ファイル感染、コード注入)

- ファイル感染



- コード注入

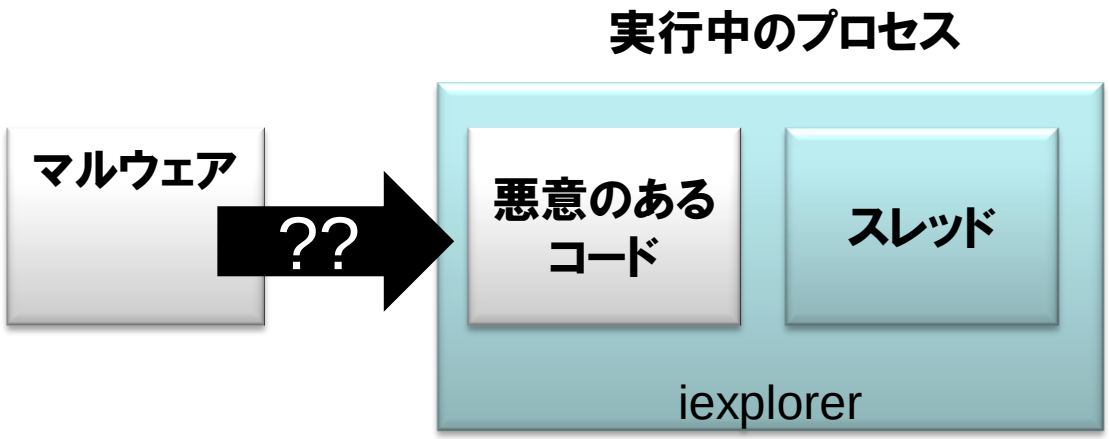


問題定義

- PID、TIDによる解析対象コード識別の問題点
  - –監視の粒度が粗い
  - –未知のコード注入を追跡できない

悪意のあるコードと通常のコードが同一のPID、TID上で動作してしまう

ファイル感染の場合



コード注入方法が分からないと、どのプロセスにコードを注入したかを追跡できない

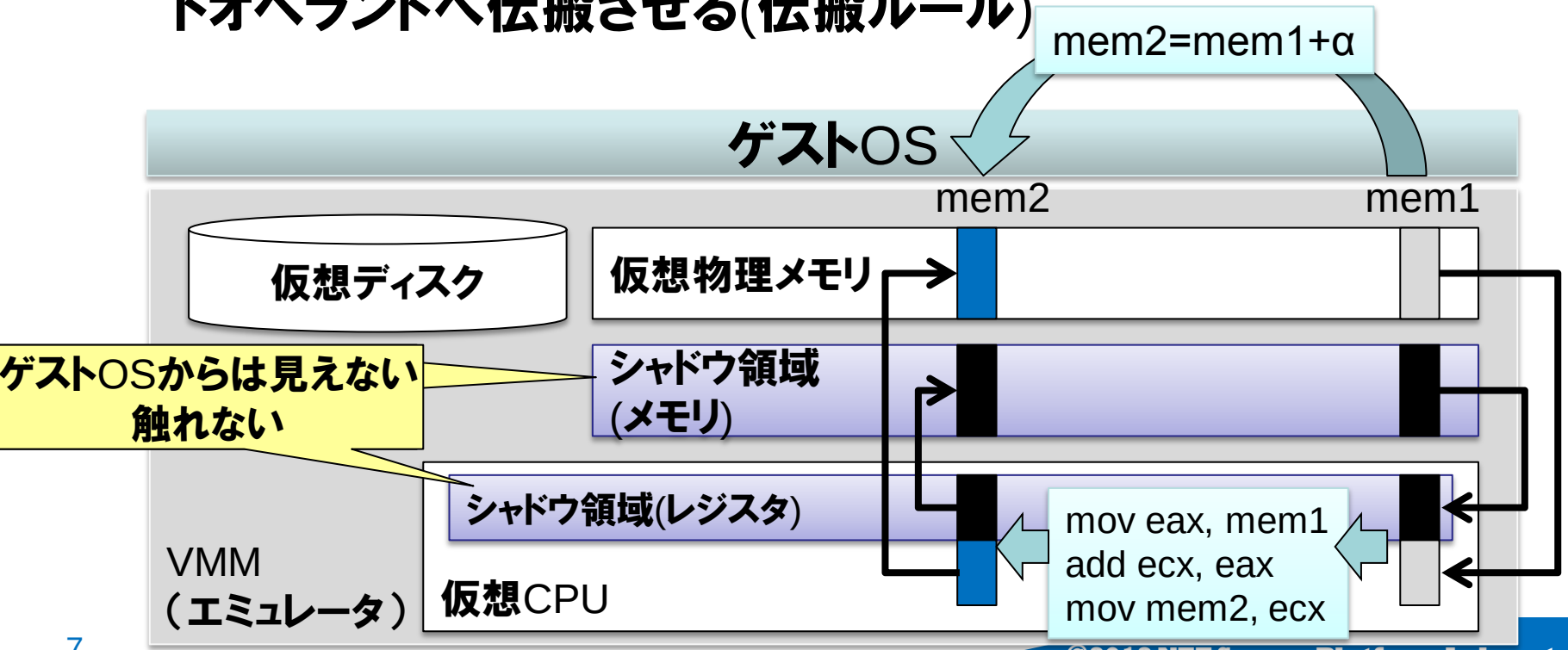
コード注入の場合

提案手法

- テイントタグに基づく解析対象識別、追跡方法
  - テイントタグを解析対象コードの設定する
    - テイントタグが付いている命令は解析対象コードとして実行
    - 付いていない命令は通常コードとして実行
  - 解析対象コードの場所が変わった場合は、テイントタグを伝搬させて追跡する

テイント解析の例

- テイントタグ(テイント解析)とは？
  - あるデータのフローを解析する技術(データフロー解析)
  - データに対して属性情報(テイントタグ)を設定する
    - テイントタグはシャドウ領域(メモリ、レジスタ)に保存する
  - CPUで実行される命令のソースオペランドのテイントタグをデストオペランドへ伝搬させる(伝搬ルール)



テイント解析機構の拡張

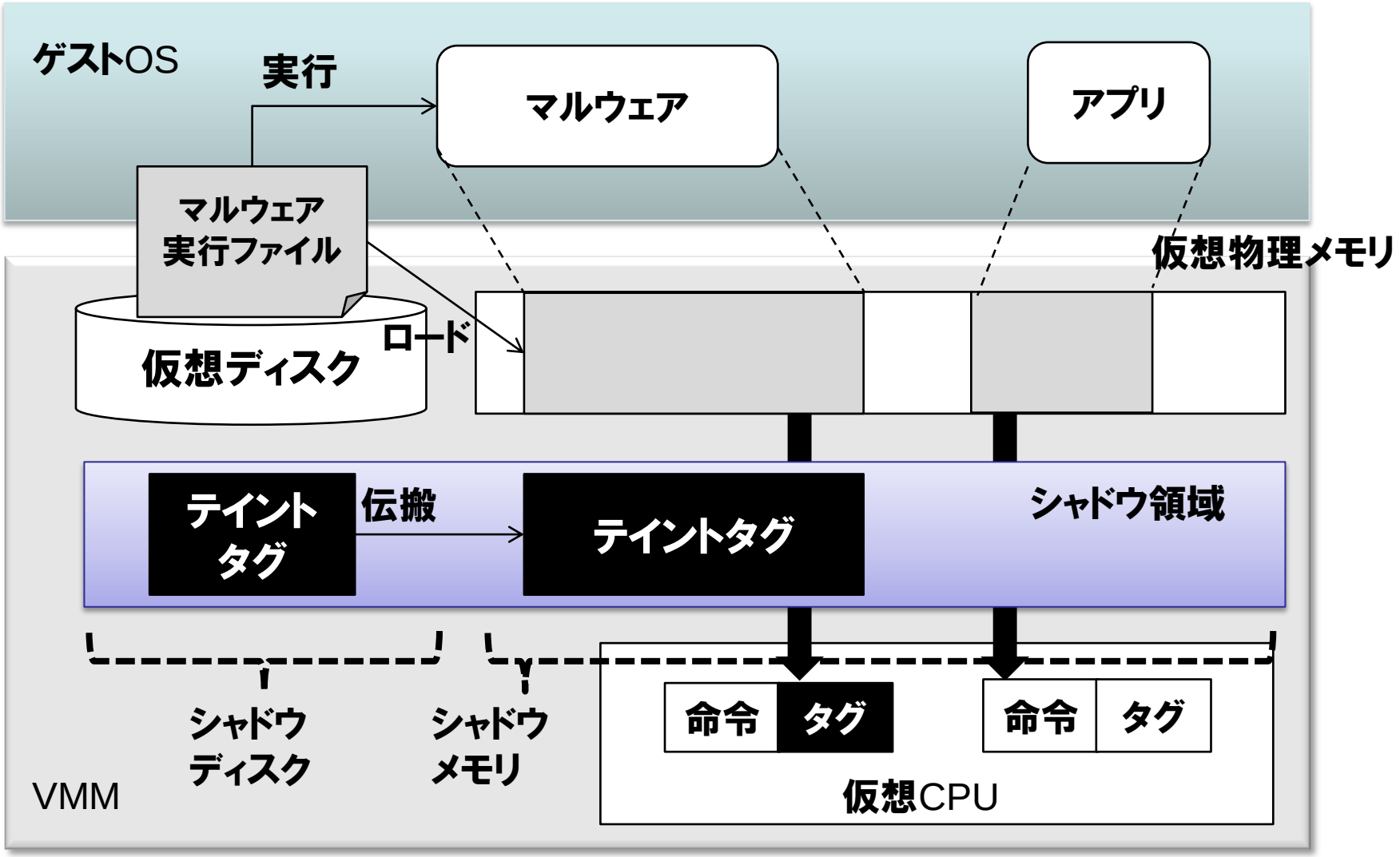
- 強制的なテイントタグ伝搬
  - 問題：マルウェアが動的にコードを生成する場合、テイントの伝搬が途切れることがある
    - データ間に直接の代入関係がない場合
  - アプローチ：解析対象コードが行った全ての書き込みに対してテイントタグを伝搬させる

```
array[0] = 0:
array[1] = 1;
…
array[255] = 255;
out = array[in]
```

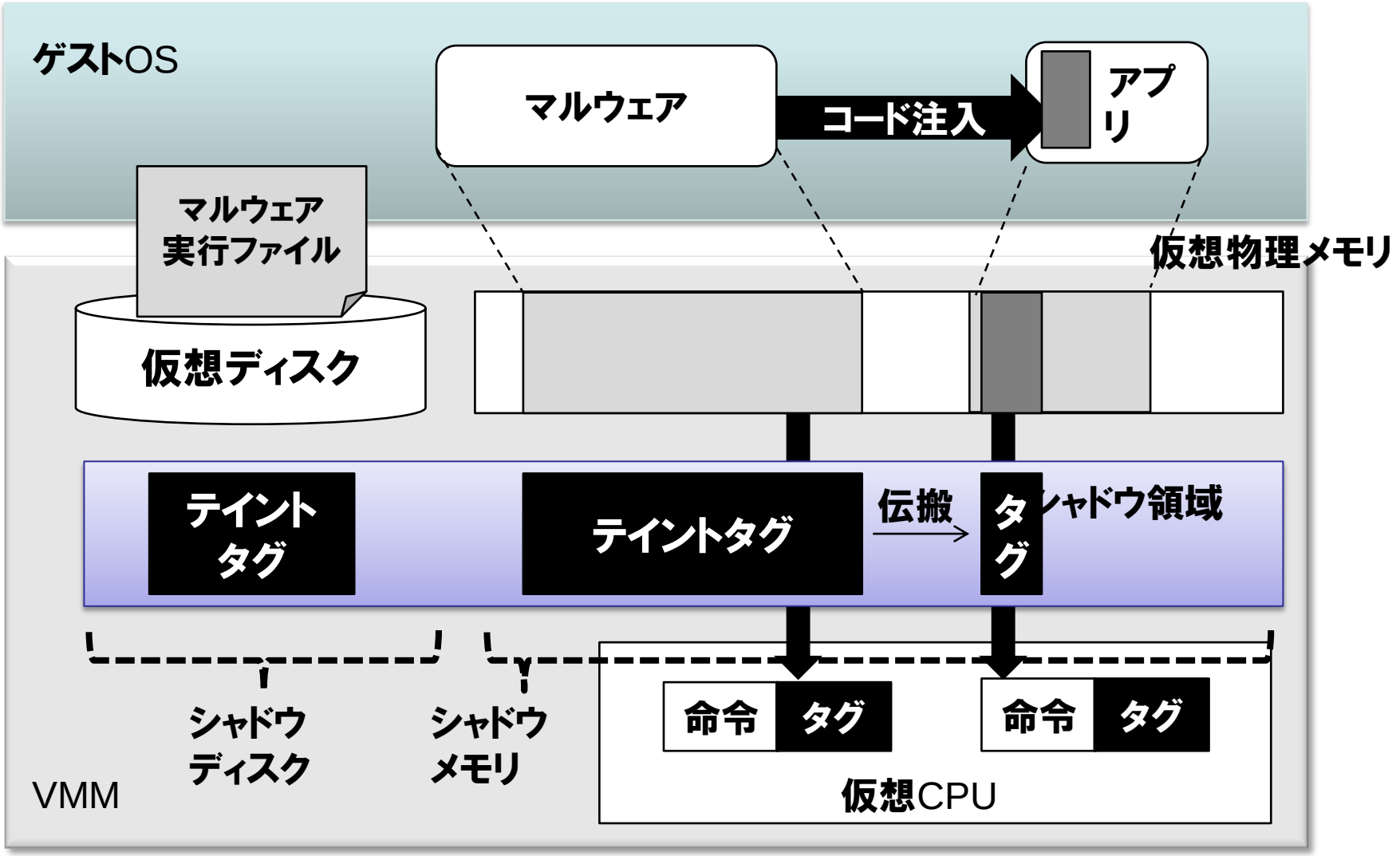
inとoutに代入関係がない例

- ファイルへのテイント伝搬
  - 問題：ファイルに書き出されるとテイントタグが消えてしまう
  - アプローチ：ディスク上のファイルへもテイントタグを伝搬させる
    - ディスク、メモリ間のデータ転送に連動して、テイントタグを転送させる

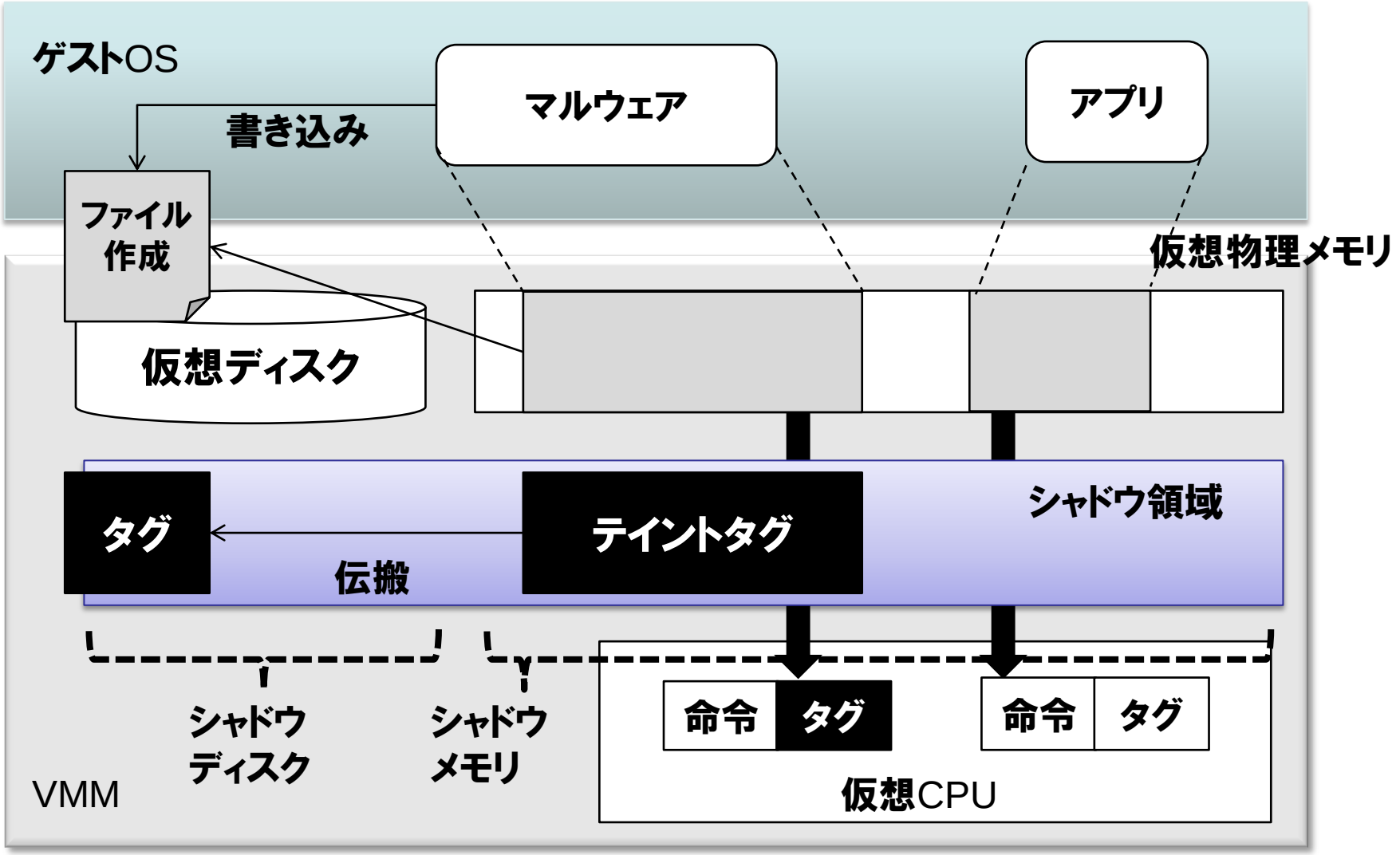
動作例(マルウェアの実行)



動作例(コード注入)



動作例(ファイル書き込み)



実装

- Argos-0.5.0を基にして実装を行った
  - ゼロデイ攻撃検知用ハニーポット
  - QEMU-0.9.1ベース
  - シャドウメモリと仮想CPUのテイント伝搬機構をそのまま利用
- 追加要素
  - 強制的なテイント伝搬
  - シャドウディスク
    - シャドウディスクへのテイント操作
    - メモリ、ディスク間のテイント伝搬機構
- ディスク上のファイル位置の特定
  - The Sleuth Kit (TSK) 3.2.3

実験

- 実験1　テストコードによる実験　合計5テストコード
  - 4種類のコード注入、1種類のファイル感染を行うコード
    - CreateRemoteThread
    - SetWindowHookEx
    - AppInit_DLLsレジストリ
    - Browser Helper Object
    - ファイル感染
- 実験2　CCC Dataset 2012による実験　32検体
  - CCC Dataset 2012からアンチウィルス名でユニークな検体　18検体
  - D3M　14検体
  - 何かしらの解析対象回避(最初の実行ファイル以外の処理が動作したのもの)をしたもの　10検体

- どちらの実験も解析対象コードに関しては、実行トレースログを取得し、手動にて誤検知、検知漏れがないかを確認した
- インターネットへの接続がない環境で、各検体を5分間実行した

実験1の結果

注入、感染方法	注入、感染先	誤検知	検知漏れ
CreateRemoteThread	calc.exe	なし	なし
SetWindowHookEx	calc.exe	なし	なし
AppInit_DLLs	user32.dllを ロードするプロセス	なし	なし
BHO	Internet Explorer	なし	なし
ファイル感染	calc.exe	なし	なし

- 誤検知（誤トレース取得） 注入、感染先のトレースを取得していないか？
- 検知漏れ（トレース取得漏れ） 注入したコード、書き出したコードのトレースを取得できているか？

誤検知、検知漏れなく正確に実行トレースを取得できた

実験2の結果

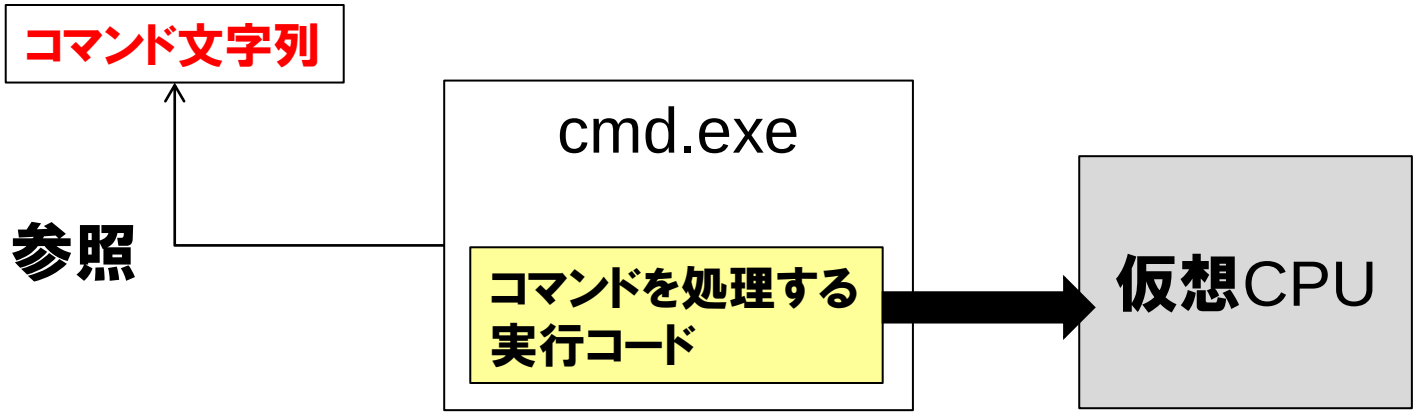
ハッシュ値	解析妨害	誤検知	検知漏れ
4cfab…	svchost.exeを改ざんして実行	なし	なし
84114…	CreateRemoteThreadでコード注入	なし	なし
493e3…	cmd.exeでバッチファイルを実行	なし	**あり**
7aaed…	cmd.exeでバッチファイルを実行	なし	**あり**
9e366…	子プロセスをサービスとして実行	なし	なし
e95f7…	子プロセスをサービスとして実行	なし	なし
28d7c…	子プロセスをサービスとして実行	なし	なし
4ead4…	子プロセスをサービスとして実行	なし	なし
52a6d…	CreateRemoteThreadでコード注入	なし	なし
5452e…	WmdmPmSNを改ざんして実行	なし	なし

493e3、7aaed以外は
誤検知、検知漏れなく実行トレースを取得できた

※CreateProcessで子プロセスを作るだけのものは除外

実験2の検知漏れの原因

- マルウェアがバッチファイルを作成し、それをcmd.exeで実行している
  - CreateProcessW(…, "C:¥WINDOWS¥system32¥cmd.exe" /c "C:¥DOCUME~1¥ADMINI~1¥LOCALS~1¥Temp¥POS1.tmp.BAT“, …)
    - POS1.tmp.BAT…tainted
    - cmd.exe …non-tainted



仮想CPU上で実行されるのが、コマンド文字列(tainted)自体でなく、そのコマンドを処理するcmd.exe(non-tainted)のコード

考察

- 制限（Limitation）
  - テイント伝搬の正確性
    - テイントの伝搬漏れ
      - 特定のAPI（暗号系、文字コード変換系）
      - スクリプト
    - テイントの誤伝搬
      - 意図的な誤検知（強制伝搬の副作用）
- 今後の課題
  - スクリプト（インタープリタ、cmd.exe）
    - taintedなデータがコードフローに影響する場合の対処法
    - 静的解析との組み合わせ
  - テイント伝搬の正確性の向上
      - DTA++、Pointer Tainting

関連研究

- TTAnalyze
  - **ゲストOS内にモジュールをインストールし、ゲストOSのPIDを取得する**
- Panorama
  - TTAnalyze**同様、ゲストOS内にモジュールをインストールし、**PID/TID**を取得する**
- Ether
  - PEB**の位置を特定し、**PID、TID**を取得する**
- VMwatcher
  - Windows**の各種データ構造を予め解析し、その情報に基づいて**PID**を取得する**
- Virtuoso
  - **メモリフォレンジックツール自動生成システム。一旦ゲストOS上で動作させたスクリプトの挙動を記録し、それをゲストの外から模倣させる。**

上記の動的解析システムはPID/TIDを基に解析対象コードの識別を行っている
本提案手法を利用することでこれらのシステムの精度を向上させることが可能

まとめ

- テイントタグに基づく解析対象コードを識別、追跡する手法を提案
- Argosにシャドウディスク、強制的なテイント伝搬機能を実装
- ５種類のテストコード、３２検体で実験し、正確性を検証
  - 検知漏れが発生したケースに関して調査、考察を行った

本提案手法を利用することで、既存の様々な動的解析システムの精度を向上させることができる