Vol.2022-CSEC-99 No.6
Vol.2022-SPT-49 No.6
Vol.2022-EIP-98 No.6
2022/12/22

機械学習を用いたランサムウェア検知における メモリとストレージのアクセスパターンの特徴重要度の分析

水野広基[1] 平野学[1] 小林良太郎[2]

概要：過去 10 年間でランサムウェアは世界中の様々な規模の組織に影響を与える最も破壊的な攻撃となりつつある．我々の先行研究では脆弱なオペレーティングシステムが攻撃を受けたとしても保護を継続できるようにするため，準パススルー型ハイパーバイザを用いてランサムウェアを検知する研究を行ってきた．先行研究ではストレージ装置へのアクセスパターンを収集して機械学習モデルを訓練することでランサムウェアを検知した．しかしながら最新鋭のランサムウェアは身代金要求までの時間を短縮するためにファイルの一部だけを暗号化する等の工夫をしており，ストレージ装置へのアクセスパターンを極端に発生させないものが現れてきている．そこで，本稿ではストレージ装置へのアクセスパターンに加えて，メモリ（RAM）へのアクセスパターンも同時に収集して新しいデータセットを構築する．本稿ではメモリとストレージの両方のアクセスパターンを用いた機械学習モデルによるランサムウェアの検知性能を評価する．本稿では，ハイパーバイザを用いたメモリとストレージのアクセスパターン収集の仕組みと機械学習モデルに入力する前の特徴エンジニアリングの方法を説明する．その後，ランサムウェア 6 種類と良性プログラム 6 種類を実行して構築したデータセットを用いて LightGBM で機械学習させた結果を示す．データセットは CPU が 2 種類とメモリ容量が 2 種類の環境で実行したデータが含まれるため，これらの条件がクラス分類に与える影響を分析した．最後にメモリとストレージのアクセスパターンそれぞれの特徴重要度の分析結果を示す．

キーワード：ハイパーバイザ，ランサムウェア，仮想化，メモリ，フォレンジック，機械学習

1. はじめに

コンピュータウイルスやワーム，トロイの木馬，スパイウェアなど，不正かつ有害な悪意のあるプログラムであるマルウェアは，インターネットの発展とともにその数を急速に増やしてきた．ドイツの IT セキュリティの研究機関である AVTEST が公開している過去 10 年のマルウェアの増加を示すグラフを図 1 に示す[1]．このようなマルウェアの増加からコンピュータを保護するためには，コンピュータに侵入したマルウェアを早期に検知する必要がある．マルウェアの代表的な検知手法に「パターンマッチング」と呼ばれる手法がある．パターンマッチングでは，マルウェア検体や感染したファイルに含まれるパターンをデータベースに登録しておきコンピュータをスキャンする．この時に利用されるマルウェアのデータパターンをシグネチャと呼ぶ．既知のマルウェアはシグネチャがデータベースに登録されているため確実に検知できる．パターンマッチングは既知のマルウェアに対しては非常に有効な手法であるが，未知のマルウェアに対しては効力がない．それに対して，未知のマルウェアでも検知できる検知手法のひとつに「ヒューリスティック検知」がある．ヒューリスティック検知では，パターンマッチングのようにデータそのもののパターンをチェックするのではなく，プログラムによる特定の動作を検知する．ヒューリスティック検知にはどうしても誤検知が起こる課題があるため現在も研究が続けられている．

我々は動的ヒューリスティック検知である振る舞いによる検知の研究を行ってきた．先行研究[2]では準パススルー型ハイパーバイザ BitVisor [3]を用いてゲスト OS 上で動作するランサムウェアをストレージ装置へのアクセスパターンを用いて検知する手法を検討してきた．しかし，ランサムウェアに似た振る舞いを持つ無害なプログラムの誤検知や，最近の OS で導入されている暗号化ファイルシステム，さらにメモリ上のみで活動するファイルレスマルウェアの登場などによって，ストレージ装置へのアクセスパターンのみによる検知が難しくなってきている．コンピュータ上で動作するプロセスは必ずメモリ（RAM）に展開されるため，前述のストレージ装置のアクセスパターンからの検知が困難な状況であっても，メモリへのアクセスパターンにある有用な振る舞いの特徴が利用できると考えた．そこで先行研究[4]において，メモリ（RAM）へのアクセスパターンを収集する監視システムを開発した．

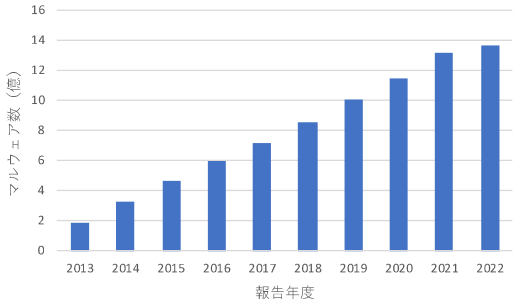

図 1 AVTEST に報告されたマルウェアの数の推移[1]

本稿ではストレージのアクセスパターンに加えて，先行研究[4]を用いてメモリ（RAM）のアクセスパターンも収集して新しいデータセットを構築し，そのデータセットで機械学習モデルを訓練してランサムウェアを検知できるかを

1 豊田工業高等専門学校 専攻科 情報科学専攻
2 工学院大学 情報学部

Vol.2022-CSEC-99 No.6
Vol.2022-SPT-49 No.6
Vol.2022-EIP-98 No.6
2022/12/22

検証する．特に，データセットは CPU が 2 種類とメモリ容量が 2 種類の環境で実行したデータが含まれるため，これらの条件がクラス分類に与える影響を分析する．さらにメモリとストレージのアクセスパターンそれぞれの特徴重要度についても分析する．

2. 仮想化技術とメモリ管理

本稿ではランサムウェアの振る舞いを仮想化技術を利用して収集するため，その概要をまずは説明する．

2.1 ハイパーバイザ

ハイパーバイザはゲスト OS より低いレイヤで動作し，ゲスト OS に対して仮想的にハードウェア資源を提供することで，複数 OS を 1 つのコンピュータ上で動作させるソフトウェアである．ハイパーバイザは仮想化手法の違いにより，ホスト型ハイパーバイザ（TYPE2）とベアメタル型ハイパーバイザ（TYPE1）に分けられる．ホスト型ハイパーバイザは OS 上にインストールされたハイパーバイザがエミュレータとしてゲスト OS を動作させる．つまりハードウェア上ではホスト OS が動作し，デバイスドライバやメモリ管理などのホスト OS が提供する機能を利用できる．ホスト型ハイパーバイザは容易に導入できるがハードウェア資源へのアクセスにホスト OS を介するためオーバーヘッドが生じる．それに対してベアメタル型ハイパーバイザはホスト OS を必要とせず，ハイパーバイザが直接ハードウェア上で動作する．そのためデバイスドライバやメモリ管理といった OS の基本機能はベアメタル型ハイパーバイザが提供しなければならない．しかし，ベアメタル型ハイパーバイザはハードウェア資源へのアクセス時にホスト OS を介する必要がないためオーバーヘッドが最小化され，システムの信頼性も向上する．先行研究[2][4]及び本稿で採用する BitVisor [3]はベアメタル型ハイパーバイザである．

2.2 BitVisor

BitVisor は Intel 社の x86 CPU プラットフォーム向けの仮想化技術 VT-x で実装された軽量の準パススルー型ハイパーバイザである．準パススルー方式は仮想化対象のハードウェア資源だけを仮想化し，それ以外のハードウェア資源をゲスト OS へそのまま通過（パススルー）させることでオーバーヘッドを最小化する． このため BitVisor でランサムウェアの動作を監視することで，準パススルー方式以外のハイパーバイザを使うよりも仮想化なしに近い状態で監視できると考えられる．BitVisor 以外の多くのハイパーバイザは同時に複数のゲスト OS の実行をサポートする．BitVisor は 1 つの OS の仮想化のみに対応することで仮想化機能が複雑にならないように設計されている．この設計方針は，ハイパーバイザを Trusted Computing Base (TCB) つまりセキュリティ強制の基盤として利用する際の信頼性向上に寄与している．

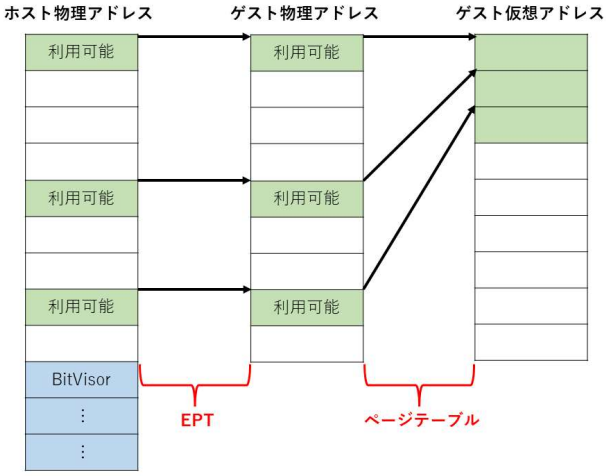

図 2　仮想化されている時のアドレス変換の仕組み

2.3 Extended Page Table

Intel 社の仮想化技術 VT-x をサポートする CPU は Extended Page Table (EPT) と呼ばれるメモリ仮想化機能を備えている[5]．仮想アドレスと物理アドレスの変換を行う通常の OS が管理するページテーブルとは異なり，EPT はゲスト OS の物理アドレスとホストの物理アドレスの変換に用いられる（図 2）．EPT でゲスト物理アドレスからホスト物理アドレスへ変換する際には 4 種類の EPT ページング構造体をページウォークすることでアドレス変換を実現する．各 EPT ページング構造体は PML4E，PDPTE，PDE，PTE と呼ばれ，それぞれ 512 GiB，1 GiB，2 MiB，4 KiB の粒度でアドレス変換を処理する．さらに EPT では EPT ページング構造体ごとに，読み込み，書き込み，実行（命令フェッチ）を許可するアクセス権限の設定を行える．そのアクセス権限に違反するアクセスが発生するか，EPT にまだエントリが登録されていない物理アドレスへのアクセスがあった場合，EPT violation による VM exit が発生し，処理がゲスト OS からハイパーバイザに遷移する．ハイパーバイザを用いてメモリ（RAM）を監視する研究の多くはこの EPT violation の仕組みを利用しており，先行研究[4]でもメモリ（RAM）へのアクセスを監視するために EPT Violation を採用している．

2.4 EPT Violation と Translation Lookaside Buffer

EPT violation を利用してメモリ（RAM）アクセスを監視する場合，Translation Lookaside Buffer (TLB) と呼ばれるキャッシュの存在を考慮する必要がある．TLB は CPU に内蔵されたキャッシュの一つであり，アドレス変換をキャッシュする．EPT violation は EPT にエントリがない場合や権限がない場合に発生するが，過去のアドレス変換結果が TLB にキャッシュされていると，そのアドレスへのアクセスで EPT violation が発生しなくなり，メモリアクセスを監視できなくなる．TLB は CPU に内蔵されているためソフ

Vol.2022-CSEC-99 No.6
Vol.2022-SPT-49 No.6
Vol.2022-EIP-98 No.6
2022/12/22

トウェアの仲介を必要とせず，自動的にアドレス変換をキャッシュする．本研究では深井らがベアメタルクラウドのライブマイグレーションを実装した際に用いた TLB shootdown を採用し[6]，すべてのコアの TLB を指定したタイミングでクリアできるようにした．

3. 提案システム

本研究で用いるメモリアクセスパターンの収集機能[4]は BitVisor [3]に C 言語で実装されている．収集したデータセットを機械学習させる際には Python と Scikit-learn 0.23.2 を利用した．それぞれの詳細を以下に説明する．

3.1 ハイパーバイザへのデータ収集機能

本稿では先行研究[2]のストレージアクセスパターン収集機能と先行研究[4]のメモリアクセスパターン収集機能を同時に使用してデータセットを構築した．メモリとストレージのアクセスパターン収集機能を図 3 に示す．ストレージアクセスパターンの収集については先行研究[2]の方法で行う．具体的にはハイパーバイザの AHCI (Advanced Host Controller Interface) 準パススルードライバを改良して Direct Memory Access (DMA) による SSD と OS との間の入出力を監視する．メモリアクセスパターンの収集は先行研究[4]をベースに行う．メモリアクセスパターンの収集には EPT violation を用いる．具体的には EPT にエントリが登録されていない物理アドレスへのアクセスで EPT violation が発生した際に VM Exit により処理がゲスト OS から BitVisor に遷移することを利用し，そのタイミングでメモリアクセスの種類と物理アドレスを取得する．Write アクセス時は先頭から非ゼロのバイトデータを 4 KiB 分，あとでエントロピーを計算するために取得する．本稿ではこれらを時系列にまとめたものをアクセスパターンと呼ぶ．EPT violation が発生するたびにゲスト物理アドレスとホスト物理アドレスの対応づけが EPT に登録されていくため，時間が経つと EPT violation の発生回数が減っていく．本稿のシステムでは EPT violation が発生しない限りアクセスパターンを収集できないため，データ収集開始時に EPT と TLB をクリアしている．

3.2 特徴量エンジニアリングと機械学習

本研究では収集したメモリとストレージのアクセスパターンから特徴量を計算し，機械学習モデルを訓練することで，最終的にランサムウェアか良性プログラムかを判別する．本稿の機械学習アルゴリズムには勾配ブースティングを用いた決定木アルゴリズム Light Gradient Boosting Machine (LightGBM) を採用した．本稿で使用する特徴量を表 1 に示す．表 1 の 23 次元の特徴ベクトルを先行研究[2]に示す時間ウィンドウをシフトさせていく方法で 10 秒単位で作成し，それぞれに正解ラベル（2 クラス分類ならランサムウェアか良性プログラムか，12 クラス分類なら検体名かプログラム名）を付与して訓練データとした．なお，表 1 のグレーの特徴量はデータセット構築時にアクセスが観測されなかったため実際には特徴量として寄与していないと考えられる．

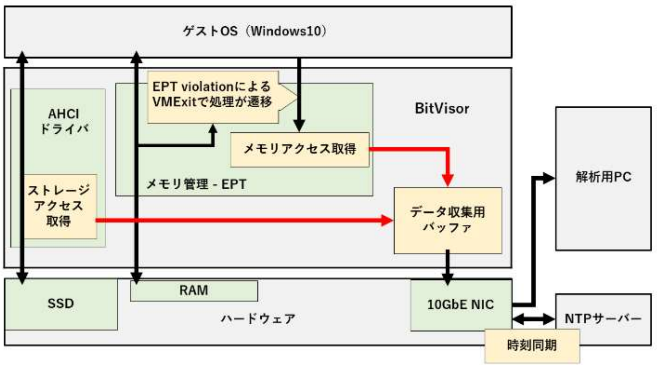

図 3　メモリとストレージのアクセスパターン収集機能

表 1　抽出する特徴量

ストレージ	Write	書き込んだデータのエントロピー
		書き込んだデータ量
		書き込んだ LAB の分散
	Read	読み込んだデータ量
		読み込んだ LBA の分散
メモリ	Write	書き込んだデータのエントロピー
		4K EPT violation の回数
		2M EPT violation の回数
		MMIO EPT violation の回数
		書き込んだ物理アドレスの分散
	Read	4K EPT violation の回数
		2M EPT violation の回数
		MMIO EPT violation の回数
		読み込んだ物理アドレスの分散
	Exec	4K EPT violation の回数
		2M EPT violation の回数
		MMIO EPT violation の回数
		実行した物理アドレスの分散
	Read-write	読み書きのデータのエントロピー
		4K EPT violation の回数
		2M EPT violation の回数
		MMIO EPT violation の回数
		読み書きした物理アドレスの分散

4. 実験

ランサムウェアは WannaCry，Conti v3，REvil，Ryuk，Darkside，LockBit の 6 種類，良性プログラムには Windows 10 標準の Zip プログラムによるファイル圧縮，暗号化プログラム AESCrypt によるファイル暗号化，データ消去プログラム SDelete によるファイル消去，Web ブラウザ Firefox の自動巡回，Office（Word，Excel，PowerPoint）のマクロ実行，Windows 10 起動後に何もしない状態の 6 種類のデータセットを構築した．

ランサムウェアのアクセスパターン収集に用いたマシンのハードウェア仕様を表 2 に示す．今回の実験では CPU

Vol.2022-CSEC-99 No.6
Vol.2022-SPT-49 No.6
Vol.2022-EIP-98 No.6
2022/12/22

を 2 種類，RAM を 2 種類用意し，計 4 パターンの環境でランサムウェアを実行させてアクセスパターンを収集した．ゲスト OS には Windows 10 LTSC を使用した．表 2 に示した 4 種類の実行環境でアクセスパターンを実行開始から 60 秒間収集した．実行環境ごとに 12 検体を 10 回ずつ実行し，合計 120 回分を 4 つの実行環境ごとに計 480 回分のアクセスパターンを集めた．以降では構築したデータセットによる機械学習の方法を説明する．

4.1 実験（1）実行環境ごとのクラス分類，メモリのみ・ストレージのみ・メモリとストレージ両方を用いたクラス分類

機械学習では 12 クラス分類（ランサムウェア 6 種類と良性プログラム 6 種類）と 2 クラス分類（ランサムウェアまたは良性プログラム）を行う．実験では実行環境毎のアクセスパターン（4 通り）と全ての実行環境のアクセスパターンの計 5 通りでの機械学習モデルを訓練する．12 クラス分類の実験においてはストレージ装置へのアクセスパターンのみ，メモリ（RAM）へのアクセスパターンのみ，ストレージとメモリの両方のアクセスパターンを使用の 3 通りで機械学習させた時の性能を調査する．2 クラス分類ではストレージとメモリの両方のアクセスパターンを使用した場合の性能を評価する．データセットは訓練用とテスト用にランダムに 50%ずつ分割して使用する．

4.2 実験（2）異なる実行環境のデータセットで訓練した機械学習モデルによる２クラス分類

提案手法が実行環境（ハードウェア）に依存していないかを検証するため，異なる実行環境で収集したアクセスパターンで訓練した機械学習モデルが，他の実行環境で収集したアクセスパターンを正しく分類できるかを調査する．表 2 の実行環境①で収集したアクセスパターン（ストレージとメモリの両方）で機械学習モデルを訓練し，他の 3 つの実行環境のアクセスパターンを 2 クラス分類させる．次に，3 つの実行環境で収集したアクセスパターン（ストレージとメモリ両方)をまとめて訓練した機械学習モデルで，残り一つの実行環境のアクセスパターンを 2 クラス分類する。この実験を 4 通りで行う．

5. 結果

実験（1）と実験（2）の結果を以下に示す．その後に実験（1）と実験（2）の実験で用いた決定木から算出された特徴重要度（feature importance）を報告する．

5.1 実験（1）実行環境ごとのクラス分類，メモリのみ・ストレージのみ・メモリとストレージ両方を用いたクラス分類

実行環境ごとのアクセスパターンを独立した機械学習モデルとして訓練して 12 クラス分類を行った時の F 値を表 3 に示す．全ての実行環境のアクセスパターンをまとめて訓練した F 値も一緒に示す．表 3 を見るとストレージ装置のアクセスパターンのみ学習した場合と比べ，ストレージとメモリの両方のアクセスパターンの方が高い F 値になった．故に，先行研究[2]の検知性能を向上させるための新たな特徴量としてメモリ（RAM）のアクセスパターンを用いる方針が正しかったことを確認できた．表 3 で実行環境ごとの F 値に着目すると，それぞれの実行環境すなわち CPU と RAM 容量の違いがランサムウェア検知の性能に大きな影響を及ぼさないことを確認した．一方，4 種類の実行環境をまとめて学習した場合は，個別に学習した場合より僅かに性能が低下した．この結果から，実行環境ごとに固有の特徴があり，実行環境を個別に学習したときにはその特

表 2　データ収集に用いたマシンのハードウェア仕様

実行環境	ハードウェア	規格
実行環境① CPU：低速 RAM：大	CPU	Intel Celeron G3920 (Gen6) 2.9GHz (2 Core) 最大メモリ帯域幅 34.1 GB/s メモリチャネル最大 2
	RAM	DDR4-2133, 2x8GB (16GB) (PC4-17000, 17GB/s x 2)
	マザーボード	ASRock H110M-HDV
実行環境② CPU：低速 RAM：小	CPU	Intel Celeron G3920 (Gen6) 2.9GHz (2 Core) 最大メモリ帯域幅 34.1GB/s メモリチャネル最大 2
	RAM	DDR4-2133，8GB (PC4-17000, 17GB/s)
	マザーボード	ASRock H110M-HDV
実行環境③ CPU：高速 RAM：大	CPU	Intel i3 12100 (Gen12) 基本 3.3/最大 4.3GHz (4 Core / 8 Thread) 最大メモリ帯域幅 76.8GB/s メモリチャネル最大 2
	RAM	DDR4-3200，2x8GB (16GB) (PC4-25600, 25.6GB/s x 2)
	マザーボード	ASRock B660M-HDV
実行環境④ CPU：高速 RAM：小	CPU	Intel i3 12100 (Gen12) 基本 3.3/最大 4.3GHz (4 Core / 8 Thread) 最大メモリ帯域幅 76.8GB/s メモリチャネル最大 2
	RAM	DDR4-3200，8GB (PC4-25600, 25.6GB/s)
	マザーボード	ASRock B660M-HDV

表 3　実行環境ごとの 12 クラス分類の F 値

	ストレージのみ	メモリのみ	両方
実行環境①	0.936	0.943	0.98
実行環境②	0.934	0.953	0.981
実行環境③	0.949	0.959	0.986
実行環境④	0.924	0.958	0.983
全実行環境	0.902	0.929	0.979

Vol.2022-CSEC-99 No.6
Vol.2022-SPT-49 No.6
Vol.2022-EIP-98 No.6
2022/12/22

徴にフィットした学習がなされていることが推測される．

次に表 4 に 2 クラス分類（ランサムウェアか良性プログラムか）の F 値を示す．2 クラス分類ではストレージとメモリの両方のアクセスパターンでのみ機械学習をおこなった．実行環境毎の F 値は 0.989 から 0.993 となり，表 3（12 クラス分類）と同様に実行環境の違いが大きく影響を与えないことを確認した．また，4 種類の実行環境をまとめて学習した場合は個別に学習した場合より僅かに F 値が低くなる結果となり，12 クラス分類と同様の傾向を示した．

5.2 実験（２）異なる実行環境のデータセットで訓練した機械学習モデルによる２クラス分類

表 2 の実行環境①で収集したアクセスパターン（ストレージとメモリ両方）で機械学習モデルを訓練し，他の 3 つの実行環境のアクセスパターンを 2 クラス分類した結果を図 4，図 5，図 6 に示す．更にこれらの実験の F 値をまとめたものを表 5 に示す．図 4 は実行環境②のアクセスパターンを分類した結果，図 5 は実行環境③のアクセスパターンを分類した結果，図 6 は実行環境④のアクセスパターンを分類した結果である．表 5 を見ると実行環境①で学習したモデルは CPU が同じ実行環境②で F 値が高く，異なる CPU（実行環境③と④）では F 値が低かった．この結果から CPU の違いが影響を与えることがわかった．ただし，今回の実行環境では実行環境①②と③④では CPU と同時に RAM の周波数やマザーボードも異なるため，現段階では今回の結果が CPU だけの違いによるものか RAM の周波数の違いによるものか，さらに別の要因があるのかまでは確認できなかった．

図 4，図 5，図 6 は上半分がランサムウェアを，下半分が良性プログラムを示している．横軸は分類割合を示しており，ランサムウェア（上半分）では橙色が多く，良性プログラムでは青色が多くなっていることが分かる．特に図 5 と図 6（CPU が異なる実行環境）で良性プログラムのうち AESCrypt，Zip，Idle（放置）はランサムウェアと誤認識されやすかった．同様にランサムウェア Ryuk は異なる CPU で 50%程度が良性と誤分類され，ほぼ分類できない状態になったことを確認した．

次に，3 つの実行環境のアクセスパターンをまとめて学習したモデルで，残り 1 つの実行環境のアクセスパターンを 2 クラス分類した際の F 値を表 6 に示す．表 6 を見ると全体として表 5 より汎化性能の高いモデルになっている．さらに，先ほど示した図 4 の結果から実行環境①と実行環境②はアクセスパターンが似ていると思われるが，実行環境②のアクセスパターンを分類するとき，①でのみ学習したモデルよりも①③④まとめて学習したモデルの F 値が高かった結果を踏まえると，単一の実行環境のアクセスパターンで学習するよりも複数の実行環境のアクセスパターンで学習したほうが良いモデルになることが分かった．

表 4　実行環境ごとの 2 クラス分類の F 値

実行環境①	0.993
実行環境②	0.992
実行環境③	0.993
実行環境④	0.989
全実行環境	0.984

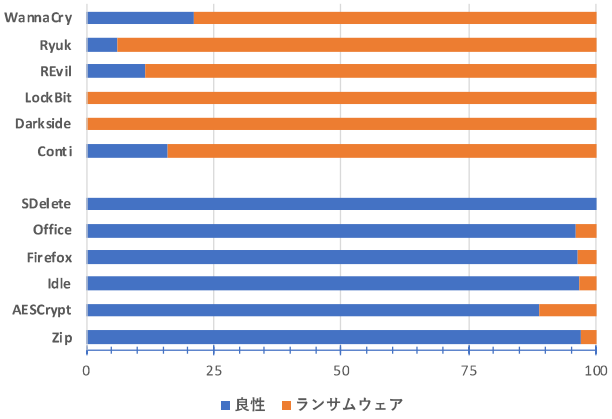

図 4　実行環境①のモデルで実行環境②を分類した結果

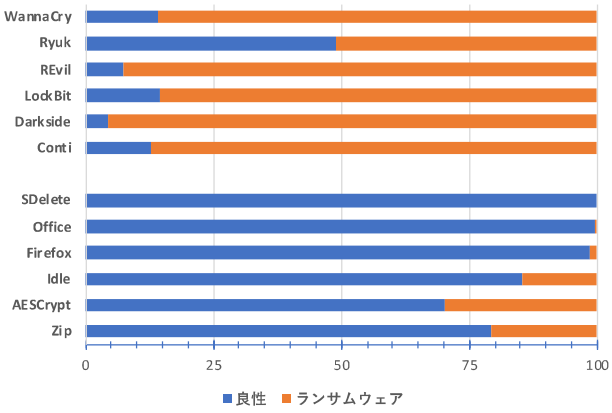

図 5　実行環境①のモデルで実行環境③を分類した結果

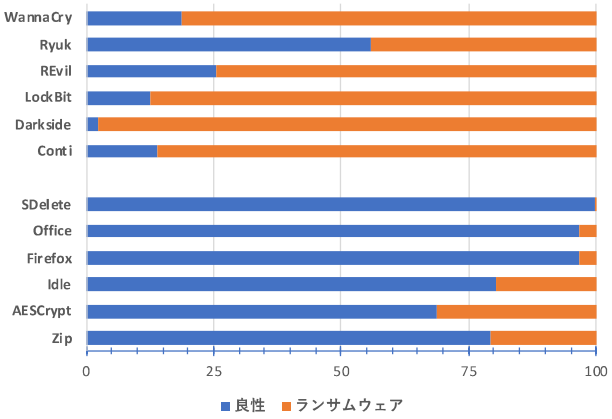

図 6　実行環境①のモデルで実行環境④を分類した結果

Vol.2022-CSEC-99 No.6
Vol.2022-SPT-49 No.6
Vol.2022-EIP-98 No.6
2022/12/22

表 5　単一の実行環境で学習したモデルの F 値

学習対象	分類対象	F値
実行環境①	実行環境②	0.932
	実行環境③	0.854
	実行環境④	0.820

表 6　3 種の実行環境で学習したモデルの F 値

学習対象	分類対象	F値
②，③，④	実行環境①	0.953
①，③，④	実行環境②	0.955
①，②，④	実行環境③	0.947
①，②，③	実行環境④	0.923

5.3 特徴重要度

実験（1）と実験（2）では決定木アルゴリズムである LightGBM を利用したため，各特徴量の特徴重要度（feature importanve）が得られた（図 7）．図 7 を見ると特徴重要度が 1%に満たない項目が 5 つあるが，これらはデータ数が 0 または極端に少なかったためと考えられる．特徴重要度が高い項目を見ると，ストレージへの書き込み量が最も寄与度が高かった．また，全体的にストレージに関係する特徴重要度はメモリに関係する特徴重要度よりも高い傾向があった．また，メモリに関しては Write に関するものが比較的重要であることが確認できた．12 クラス分類と 2 クラス分類を比較すると，ストレージへの書き込み量と読み込み量が 2 クラス分類においてより重要であることが確認できたが，全体的な傾向は大きく変化しなかった．ストレージとメモリの特徴重要度の割合は 44 : 56 であり，極端に割合の低い 5 つを除いた特徴重要度の平均はストレージが約 8.8%，メモリが約 4.3%であった．

特に特徴重要度の高かった 9 つの部分依存（Partial dependence）プロットを図 8 に示す．部分依存プロットは一つの特徴量だけを変化させ，他の特徴量を固定した時の分類結果を可視化したものである．入力する特徴量とクラス分類の定量的な関係を求めるのに用いられる．ただし，この手法は特徴量同士が独立であることが前提であり，特徴量同士に相関がある場合にはうまく機能しない可能性があることに注意を要する．今回はメモリの特徴量同士で相関関係があると考えられるためあくまで参考とする．図 8 のグラフは横軸が特徴量の数値であり，縦軸はランサムウェアを 1，良性プログラムを 0 として算出した値である．例えば，縦軸の値が 0.5 であれば，その時の特徴量の値でランサムウェアと良性プログラムの分類が半々ということになり，クラス分類に寄与していないことが分かる．例えば図 8 の右上にあるストレージの書き込み量に関する特

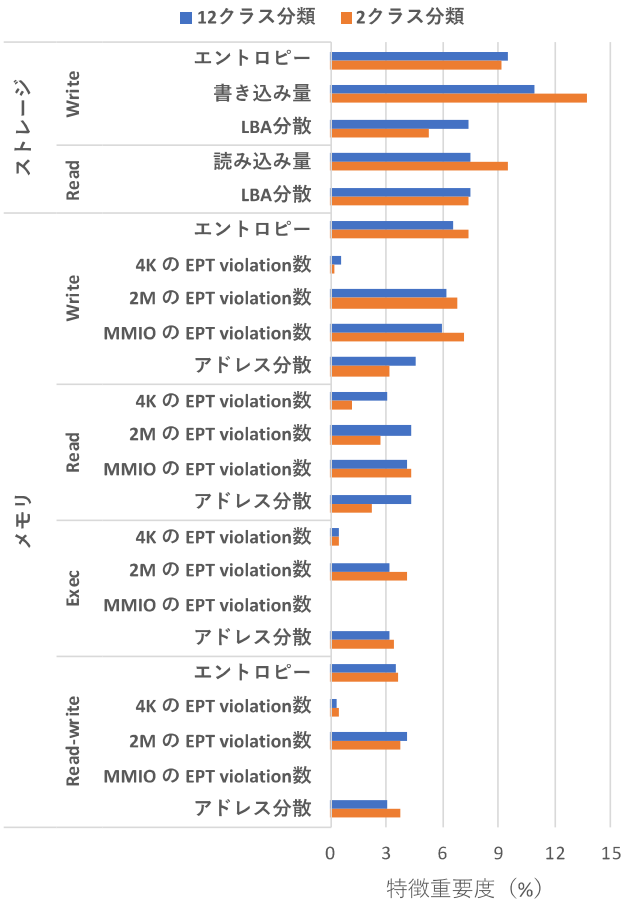

図 7　LightGBM から特徴重要度の比較

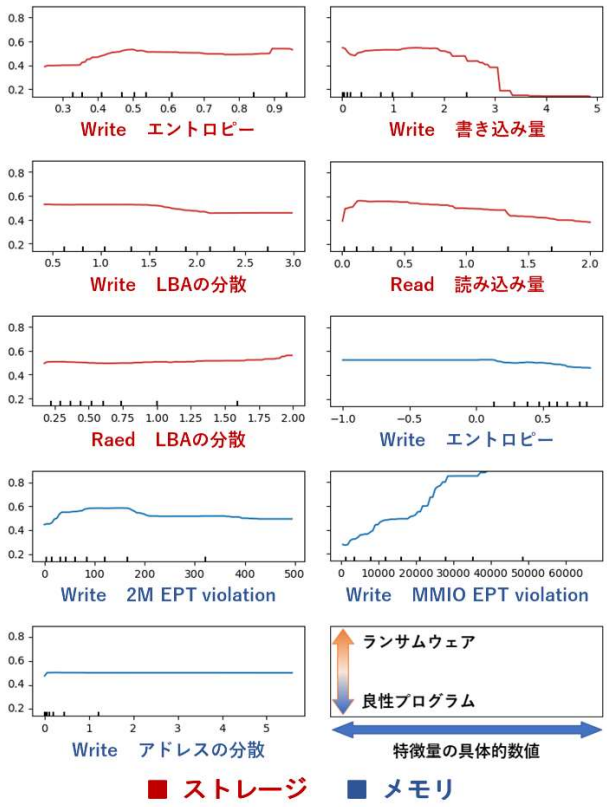

図 8　特徴量の部分依存プロット

Vol.2022-CSEC-99 No.6
Vol.2022-SPT-49 No.6
Vol.2022-EIP-98 No.6
2022/12/22

徴量では，書き込み量が一定以上の値になると良性プログラムと判断されやすいということが読み取れる．図 8 よりストレージの読み込み量は少なめであり，ストレージの書き込み量は極端に多くなく，かつ Write の MMIO のメモリアクセスが多いものがランサムウェアである確率が高いことが分かった．

6. 考察

実験（1）の結果から，ストレージ装置へのアクセスパターンだけで学習するより，メモリ（RAM）のアクセスパターンを併せて学習することで，より高い性能でランサムウェアを検知できることが確認できた．この事実は実行環境に大きく左右されないことも実験結果から確認できた．一方，実験（2）では異なる実行環境のアクセスパターンで訓練した機械学習モデルで他の実行環境のアクセスパターンをクラス分類したときに性能が低下する問題が明らかになった．複数の実行環境でアクセスパターンを収集していくことで汎化性能を高められるものの，現実に存在する PC の実行環境は無数にあるため，実行環境に左右される検知システムは望ましくない．今回の実験で CPU または RAM の周波数が異なる時に F 値が低下するという結果が得られた．今後は CPU と RAM 周波数のどちらが大きな影響を与えているのか，もしくは何か他の原因があるのかを調査し，実行環境に左右されにくい検知システムを検討していく必要がある．

特徴重要度については，ストレージのアクセスパターンでの特徴重要度の分析は先行研究[2]で示していたが，メモリのアクセスパターンに関する分析は本稿で初めて行ったものである．今回の実験では特徴重要度の高い項目が判明したが，ほとんど機能していない特徴量も見つかった．今後は，現行の 23 次元の特徴量からいくつか除外するなどして，特徴重要度の低い特徴量でも地味ながら効いているのか，もしくは足を引っ張っているのか，詳細な検証を行いたい．今回の実験では，ストレージの読み込み量は少なめであり，ストレージの書き込み量は極端に多くなく，かつ Write の MMIO のメモリアクセスが多いものがランサムウェアである確率が高いという結果が出た．ただし，これについても今回使用したランサムウェアは 6 種類だけであり，実際に存在するランサムウェア全体の傾向とは言い切れない．本システムはデータセットからルールを自動的に学習する仕組みであることから，必ずしも人間がランサムウェアのアクセスパターンがどのようなものかを定義する必要はないが，システムの改善やランサムウェアの解析に応用できる可能性を考え，更なる分析を行いたい．

7. 関連研究

Oz らはサーベイ論文[7]にてランサムウェアの進化と防御機構を体系的に調査した．Oz らの調査によると PC 向けのランサムウェア検知に関する研究論文の 73%が機械学習を採用しており，特徴量としては API コールの利用（22%），ファイルやディレクトリの特徴（20%），以下レジストリ（11%），エントロピー（8%），文字列（8%）が多用されていた．これらの多くの研究論文では OS で特徴量を得ているが，本研究ではハイパーバイザで特徴量を得ている点が異なっている．

8. まとめ

本稿では先行研究[2]のストレージ装置へのアクセスパターンを用いたランサムウェア検知に，先行研究[4]で提案したメモリ（RAM）のアクセスパターン収集システムを組み合わせて，ランサムウェアのストレージとメモリの両方のアクセスパターンを収集してデータセットを構築した．実験では実行環境の違いによるランサムウェア検知性能を評価し，最後に特徴重要度を分析した．全体としてストレージのアクセスパターンに加えて，新たにメモリ（RAM）のアクセスパターンを追加することでランサムウェアの検知性能を向上させることに成功した．しかし，アクセスパターンを収集する実行環境の影響，学習に使用するメモリの特徴量の妥当性など，更なる検証と改善の余地が存在することが分かった．今後は，まず実行環境の違いによる性能の低下が具体的に何によって引き起こされているのかを検証していく予定である．

謝辞 本研究は科研費（JP20K11825）助成を受けたものである．

参考文献

[1] AVTEST: The Independent IT-Security Institute, https://www.av-test.org/en/statistics/malware/, 2022 年 7 月 18 日閲覧.
[2] M. Hirano, R. Hodota, and R. Kobayashi. RanSAP: An open dataset of ransomware storage access patterns for training machine learning models. Forensic Science International: Digital Investigation, 40, 2022, 301314.
[3] Shinagawa, T. et al. BitVisor: a thin hypervisor for enforcing I/O device security. In Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 121-130, 2009.
[4] M. Hirano and R. Kobayashi, Machine Learning-based Ransomware Detection Using Low-level Memory Access Patterns Obtained from Live-forensic Hypervisor, Proc. of IEEE International Conference on Cyber Security and Resilience (CSR), 2022, pp. 323-330.
[5] Intel Corporation, Intel® 64 and IA-32 architectures software developer's manual. Volume 3: System Programming Guide, Nov. 2020.
[6] Fukai, T., Shinagawa, T., and Kato, K. Live migration in bare-metal clouds. IEEE Transactions on Cloud Computing, 2018.
[7] Oz, Harun, et al. A survey on ransomware: Evolution, taxonomy, and defense solutions. ACM Computing Surveys (CSUR) 54.11s, 2022, pp. 1-37.