悪性PowerShell検知のための機械学習モデルに対する回避攻撃の可能性の検証

目澤 勇樹[1,a)]　三村 守[1,b)]

概要：サイバー攻撃において，Windows に標準搭載されている PowerShell は，攻撃者に好都合なツールとなっている．先行研究では，自然言語処理技術と機械学習モデルを組み合わせ，PowerShell を分類する手法が提案されている．機械学習については，敵対的な入力による精度の低下が指摘されているが，PowerShell の分類を対象とした検証は報告されていない．そこで本研究では，悪性 PowerShell 検知のためのモデルに対しても，回避攻撃が可能であるかを検証した．実験では，先行研究の Bag-of-Words，Latent Semantic Indexing (LSI) および Support Vector Machine に加え，Doc2Vec，RandomForest および XGBoost を追加し，組み合わせごとの結果を比較した．その結果，すべての組み合わせで再現率が低下し，PowerShell においても回避攻撃が可能であることを確認した．特に，Doc2Vec を用いたモデルは他のモデルよりも攻撃の影響を受けやすく，再現率が最大で 0.78 も低下した．その影響は主に自然言語処理技術に依存しており，LSI では機械学習モデル毎の差がほとんど認められなかった．

キーワード：PowerShell，回避攻撃，潜在意味インデックス，Doc2Vec，XGBoost

Evaluating the possibility of evasion attacks to machine learning-based models for malicious PowerShell detection

Yuki Mezawa[1,a)]　Mamoru Mimura[1,b)]

Abstract: In cyber attacks, PowerShell has become a convenient tool for attackers. A previous study proposed a classification method for PowerShell scripts that combines natural language processing techniques and machine learning models. Although it has been pointed out that the accuracy of machine learning is degraded by adversarial input, no verification has been reported for PowerShell classification. In this study, we evaluated the possibility of evasion attacks to the machine learning-based model for malicious PowerShell detection. In addition to Bag-of-Words, Latent Semantic Indexing (LSI), and Support Vector Machine (SVM), we combined Doc2Vec, RandomForest, and XGBoost with the previous models. As a result, we confirmed that evasion attacks are also possible in PowerShell. In particular, the models using Doc2Vec decreased the recall rate by 0.78 at maximum. The effect mainly depends on the natural language processing technique, and there was almost no difference in any machine learning models with LSI.

Keywords: PowerShell，Evasion Attack，Latent Semantic Indexing，Doc2Vec，XGBoost

1. はじめに

ファイルレスマルウェア攻撃と呼ばれる従来のウイルス対策ソフトでは検知が困難な攻撃が増加している．2020年には，前年と比較して約 10 倍に増加したとの報告もある [1]．ファイルレスマルウェアとは，実行ファイルがディスク上に保存されず，メモリ上で実行されるマルウエアである．メモリ上のデータは電源を切ると一般に消失するため，痕跡が残りにくく発見が困難である．ファイルレスマ

[1] 防衛大学校情報工学科
National Defense Academy of Japan
[a)] em60014@nda.ac.jp
[b)] mim@nda.ac.jp

ルウェアは，その動作のために Windows の正規ツールである PowerShell を利用することが多い．このような悪性 PowerShell の検知に関する先行研究としては，伝統的な自然言語処理技術および畳み込みニューラルネットワークを組み合わせた動的解析による手法が提案されている [2]．また，自然言語処理技術に着目し，機械学習モデルと組み合わせた静的解析による検知手法も提案されている [3], [4]．このような機械学習モデルを用いた手法では未知のサンプルの検知が期待できるが，その仕組みを想定した敵対的な入力によって分類精度が低下することも指摘されている [5]．先行研究で提案された手法についても同様の影響を受ける可能性があるものの，十分な検証が実施されているとは言い難い．機械学習モデルの回避に関する先行研究としては，PDF (Portable Document Format)，Portable Executable (PE)，Android，VBA マクロ等を対象とした研究が実施されている [6], [7], [8], [9]．しかしながら，悪性 PowerShell の検知モデルを対象とした検証は，現時点では報告されていない．

そこで本研究では，単語レベルの機械学習モデルによる悪性 PowerShell の検知手法 [4] に対し，検知を回避する手法を試行し，検知率が低下するかを確認した．この検知手法では，PowerShell スクリプトを単語に分割して言語モデルを構築し，特定の単語の出現頻度等の言語的特徴から悪性 PowerShell スクリプトを検知する．ここで，悪性スクリプトに対し，良性スクリプトに頻出する単語を挿入して回避を試行する [9]．さらに，自然言語処理技術と機械学習モデルの組み合わせによる影響を比較するため，自然言語処理技術として Doc2Vec，機械学習モデルとして RandomForest および XGBoost を新たに追加した．著者らが知り得る限り，Doc2Vec を含めた自然言語処理技術と，複数の機械学習モデルに対する検証は初めての試みである．

本論文の貢献は次のとおりである．

(1) 悪性 PowerShell スクリプトに対して，良性サンプルの特徴を付与することにより，その機能を維持しつつ，機械学習モデルに対する回避攻撃が可能であることを示した．
(2) 自然言語処理技術として，新たに Doc2Vec に対して良性サンプルの特徴付与による回避手法を適用し，その効果を検証した．
(3) 機械学習モデルとして，新たに RandomForest と XGBoost に対して回避手法を適用したサンプルを分類させ，その効果を検証した．
(4) Doc2Vec を用いたモデルは Bag-of-Words (BoW) および Latent Semantic Indexing (LSI) を用いたモデルよりも回避攻撃による影響を受けやすく，再現率が最大で 0.78 も低下することを確認した．
(5) 回避攻撃による影響は自然言語処理技術に依存しており，LSI では機械学習モデルごとの差がほとんど認められないことを確認した．

2. 関連研究

2.1 悪性 PowerShell スクリプトの難読化解除

マルウエアには，解析を妨害するために難読化が施されていることが多い．これは，悪性 PowerShell スクリプトの場合にも同様である．PowerShell スクリプトの難読化を解除するために，いくつかの手法が提案されている．代表的な研究としては，Ugarte らによる PowerDrive の開発が挙げられる [10]．これは，静的および動的な多段式の難読化 PowerShell の解除ツールであり，Ugarte らは実験において，セキュリティベンダおよび VirusTotal から収集した 4828 体の悪性スクリプトの内，4642 体を正確に分析した．分析に失敗した 186 体は，PowerDrive で難読化を解除できない.NET 言語に属する API を使用していた．PowerDrive は，悪性 PowerShell スクリプトを静的解析するために必要なソースコードの難読化解除を，より効率的に実施することを可能としている．Jeff White は，正規表現を用いて悪性 PowerShell の難読化を解除し，その挙動分析の結果を示した [11]．Liu らもまた，正規表現を用いた難読化解除手法を提案し，その手法を PSDEM というモデルとして提案した [12]．Li らは，抽象構文木 (AST) のサブツリーに注目した軽量な難読化解除手法を提案した [13]．Li らの手法は，平均の難読化解除に要する時間の平均は 0.5 秒以下であり，難読化スクリプトとオリジナルとの類似度を 0.5%から約 80%にまで向上させた．これらの研究は，悪性 PowerShell スクリプトの難読化を解除することを目的としている．本研究では，悪性 PowerShell スクリプトそのものの検知に注目する．

2.2 悪性 PowerShell の検知

悪性 PowerShell を検知するために，機械学習モデルを用いた手法が提案されている．Hendler らは，ディープニューラルネットワークを用いた悪性 PowerShell の検知手法を提案した [2]．これは，深層学習アーキテクチャと伝統的な自然言語処理を組み合わせた手法であり，4-CNN と 3-gram を使用している．Hendler らはまた，単語の文脈的な埋め込みを用いた悪性 PowerShell の検知手法も提案している [14]．これは，Microsoft 社の Anti-Malware Scan Interface (AMSI) をベースに難読化された PowerShell のトークンを抽出し，単語の文脈的な埋め込み (Word2Vec および FastText)，Token-Char，Convolution Neural Network (CNN)，Recurrent Neural Network (RNN) を使用する手法である．Rusak らは，AST と深層学習を組み合わせて悪性 PowerShell を検知する手法を提案した [15]．これは，テキストベースではなくコードの構造情報を利用した検知手法である．この手法では，前処理として Base64 でエ

ンコードされたスクリプトやコマンドをデコードして AST に変換したのち，PowerShell スクリプトのコーパスに基づいて各 AST ノードのルートを構築している．Jeff White は，4100 体の悪性 PowerShell を分析し，27 のグループに分類した [16]．また，分析を通じて悪性 PowerShell がどのような標的に対して，どのような手法で攻撃しているのかを明らかにした．これらの研究を踏まえ，田尻らは静的解析のみを用いた単語ベースの言語モデルによる悪意ある PowerShell の検出手法を提案した [3], [4]．これは，自然言語処理技術の一種である LSI を用いてソースコードから単語ベースの言語モデルを生成し，機械学習を用いて未知の PowerShell スクリプトを分類する手法であり，実験では GitHub 等において公開されているサンプルを用いて再現率 (Recall)0.98 を達成している．Fang らは，複数の特徴を組み合わせた悪性 PowerShell の検出手法を提案した [17]．これは，文字レベル，関数レベル，AST レベルで抽出した特徴と，FastText で抽出した意味的特徴を組み合わせた手法であり，実験では正解率 (accuracy)0.9776 を達成している．ただし，この手法は難読化された PowerShell スクリプトには対応していないため，事前の難読化解除が必要である．

このように先行研究では，自然言語処理と機械学習または深層学習を組み合わせた悪性 PowerShell の検知手法が提案されている．同時に，PowerShell スクリプトの難読化解除に関する手法も研究され，それらを組み合わせた静的解析による検知手法は高い再現率を達成している．これらの先行研究では，実際のマルウェアを使用した評価を実施している．しかしながら，攻撃者が検知手法の存在を認識しており，回避を試みる場合までは想定されておらず，堅牢性が十分に評価されているとは言い難い．そこで本研究では，攻撃者にこれらの検知手法が認識され，回避を試みる状況を想定する．

2.3 機械学習モデルの回避

機械学習モデルを用いた検知手法に対しては，検知を回避する攻撃の脅威が指摘されている．

Maiorca らは，敵対的生成ネットワーク (GAN) を用いて機械学習による PDF マルウェア検知を回避する手法を提案し，対象とした検知手法に対する高い検知回避率を示した [6]．Chen らは，CNN を用いた PE 形式のマルウェアの検知手法を回避する手法を提案した [7]．この研究では，PE ファイルにソースコードを追加することによって検知を回避させているほか，敵対的学習や敵対的サンプルの事前検出によって攻撃に対抗する手段も示した．Grosse らは Android マルウェアにおいて，良性アプリケーションの特徴を持つコードを追加することによって，マルウェアの機能を維持しつつ動的解析による検知を回避することが可能であることを示した [8]．山本らは，VBA マルウエアに良性 VBA マクロの特徴を追加することによって，マルウェアの機能を維持しつつ，機械学習モデルによる検知を回避することが可能であることを示した [9]．この研究では，自然言語処理技術である BoW および LSI と，機械学習モデルである Support Vector Machine (SVM) を組み合わせた VBA マルウエア検知モデルに対する回避攻撃を検証している．回避の手法としては，良性マクロから得られる単語レベルの特徴を VBA マルウエアに挿入している．この検証では，BoW を用いたモデルの再現率が基準の 1.5%に低下し，LSI を用いたモデルの再現率は 73%低下することが確認されている．

このように，PDF，PE，Android，VBA マクロ等のマルウェア検知手法に対する回避攻撃の脅威は指摘されている．しかしながら，悪性 PowerShell の検知モデルに対する回避攻撃の脅威はこれまでに報告されていない．Android マルウェアおよび VBA マルウエアの例のように，良性スクリプトの特徴をソースコードに付与する攻撃は，PowerShell スクリプトにおいても可能であると考えられる．したがって本研究では，機械学習を用いた悪性 PowerShell の検知モデルを，攻撃者が回避する可能性について検証する．

3. 関連技術

本節では，本研究で使用する自然言語処理技術および機械学習モデルに関して述べる．双方とも分類精度に大きく影響を及ぼすものであるため，回避手法による影響の大小を確認するためにはそれぞれの組み合わせを比較する必要がある．本研究では，自然言語処理技術として BoW，Doc2Vec および LSI，機械学習モデルとして SVM，RandomForest および XGBoost を使用した．これらの自然言語処理技術および機械学習モデルは，それぞれが異なる性質を有しており，回避の影響を評価するために有益であると考えられる．

3.1 自然言語処理技術

Bag-of-Words (BoW) は，単語の出現頻度のみを基に文書をベクトルに変換し表現する手法である．各単語の順番は考慮されない．w を単語，n を w に対応した出現頻度として表した場合，文書 d は式 (1) で表すことができる．

$$d = [(w_1, n_{w_1}), (w_2, n_{w_2}), (w_3, n_{w_3}), ..., (w_j, n_{w_j})] \quad (1)$$

この式 (1) を元に n の位置を固定することで単語 w を省略すると，d を数値で表現することができる．これにより，各文書 $\hat{d}_i$ における単語の出現数を表した文書と単語の関係は，式 (2) のベクトルで表現できる．

$$\hat{\mathbf{d}}_\mathbf{i} = (n_{w_1}, n_{w_2}, n_{w_3}, ..., n_{w_j}) \quad (2)$$

BoW を用いることで，各文書をユニークな単語数で固定された次元数のベクトルに変換することができる．

Doc2Vec は，Le らによって提案された Paragraph Vector の実装の 1 つである [18]．Paragraph Vector は，Mikolov らによって提案された単語の特徴ベクトルを生成するモデルである word2vec を拡張したものであり，単語ではなく文書の特徴ベクトルを生成するモデルである．Paragraph Vector は，入力文の長さに制限がないため，入力の長さによって処理を変える必要がなく，また構文解析を必要としない．生成された特徴ベクトルを比較することで文書同士の類似度を算出することが可能となり，併せて文書の分類を行うことができる．

Latent Semantic Indexing (LSI) は，高次元の文書群から与えられたクエリに関連する文書を見つけるために開発された自然言語処理技術であり，文書群と文書に含まれる単語の関連性を分析することができる．LSI では，BoW 等によって求めた特徴ベクトルを特異値分解によって低次元に圧縮し，関連性を分析する．BoW 等では，単語数に比例して次元数が増えるためそれに合わせて計算量も増大するが，LSI では特異値分解により低次元に圧縮するため，文書の特徴を維持しつつ計算量を削減することができる．

3.2 機械学習モデル

SVM は教師あり学習を用いるパターン認識モデルの一種であり，分類問題および回帰問題に用いられている．マージン最大化と呼ばれる各データ点からの距離が最大となる超平面を求める考えに基づき，決定境界を求めることにより学習し，各データを判別する．

RandomForest は，決定木を弱学習器とするバギングを改良したアンサンブル学習の代表的な手法であり，重複可能なランダムサンプリングによって多数の決定木を生成し，その多数決によって予測値を決定するアルゴリズムである．

XGBoost は，決定木と勾配ブースティングを組み合わせたアンサンブル学習の一種である．拡張性があり大規模データにも対応可能であるほか，値の欠損や 0 が多い疎なデータにも対応することが可能なアルゴリズムである．

4. 検証手法

4.1 概要

検証の手順を図 1 に示す．検証は大きく訓練プロセスと検知回避プロセスの 2 つに分けられる．データセットは，訓練データ，偽装データおよび攻撃データがあり，各データには良性 PowerShell と悪性 PowerShell の両方のサンプルが含まれている．訓練データとは，分類器となる機械学習モデルを訓練するためのデータ，偽装データとは，悪性 PowerShell に対し回避手法を適用するデータ，攻撃データとは，良性 PowerShell の単語レベルの特徴の抽出元となるデータである．各プロセスに共通する処理として，それぞれのデータに対して前処理として難読化解除，データクレンジングおよび分かち書きを行う．次に，訓練プロセスによって分類器となる機械学習モデルを訓練する．訓練プロセスの後に，検知回避プロセスを実行し，良性 PowerShell の単語レベルの特徴を付与した偽装データを生成する．最後に偽装データを分類させ，分類の再現率を確認する．

本手法によって生成される機械学習モデル F を回避する悪性 PowerShell のサンプル x^* は，式 (3) で表現することができる．

$$x^* = x + \delta_{y_a} \ s.t. \ F(y_t) = F(x + \delta_{y_a}) \tag{3}$$

ここで，x は通常の悪性 PowerShell，y は良性 PowerShell，δ_y は良性 PowerShell の特徴である．また t は訓練データ，a は攻撃データを表している．本研究では，攻撃者の状況を機械学習モデルの存在を認識しており，かつ，訓練データにアクセスできる状況とした．したがって，$y_a = y_t$ となる．

以下，各手順の詳細を説明する．

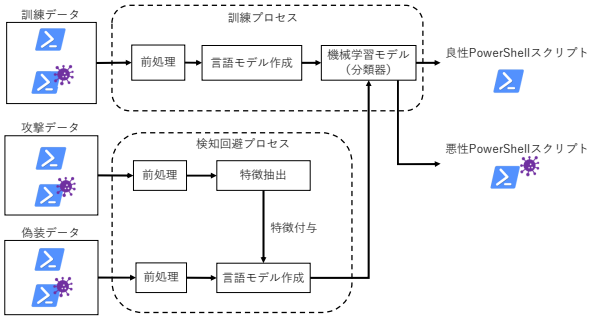

図 1 検証の手順

4.2 前処理

前処理では，各データに対して難読化解除，データクレンジングおよび分かち書きを行う．各処理は，それぞれ正規表現を利用して行う．難読化解除では，マッチングを利用して難読化部分を抽出し，小文字への統一、Base64 エンコードの置換，終端文字での改行等を行う．データクレンジングでは，コメントアウト，URL，IP アドレスおよびマルチバイト文字をそれぞれ置換処理し，特徴の一つとして利用できるようにする．分かち書きでは，コマンドや変数の境界となる記号 (空白文字，終端記号，括弧および演算子) で文字列を分割する．

4.3 訓練プロセス

訓練プロセスでは，分類器となる機械学習モデルの訓練を行う．最初に，前処理を行った訓練データに対して BoW，Doc2Vec または LSI の自然言語処理技術を適用し、言語モデルを生成する．次に，生成した言語モデルを SVM，XGBoost または RandomForest の機械学習モデルに入力

し訓練する．訓練された機械学習モデルは以後，分類器としてサンプルを分類できるようになる．

4.4 検知回避プロセス

検知回避プロセスでは，偽装データに対して回避手法を適用する．その手順を Algorithm1 に示す．最初に，攻撃データに含まれる良性 PowerShell に頻出の単語群 Y_a と悪性 PowerShell に頻出の単語群 X_a の差分を取り，良性 PowerShell にのみ頻出する単語群 δ_{Y_a} を抽出する．抽出した単語群 δ_{Y_a} から，ランダムに 1 語 δ_{y_a} を選定し，偽装データの悪性 PowerShell x に挿入する．単語を挿入する際は，PowerShell としての機能を維持させるため，大文字小文字の変換や文字数のカウント等の全体の動作に影響を及ぼさない関数の引数として使用する．ここで使用する関数 g もまた，単語を挿入するごとに事前に準備したリスト G からランダムに選択する．なお，本研究では偽装データへの挿入処理の回数 k を 0 回から 500 回の範囲で変化させ，それによる再現率の変化を測定する．全ての挿入処理が終了した後に，BoW，Doc2Vec または LSI の自然言語処理技術を適用し、言語モデルを生成する．

Algorithm 1 検知回避プロセス

Require: $X_a = \{w_{x_a n}\}$, $Y_a = \{w_{y_a n}\}$, k, $G = \{g_n\}$
  $\delta_{Y_a} = Y_a - X_a$
  while Number of adds $< k$ **do**
    $\delta_{y_a} =$ random δ_{Y_a}
    $g =$ random G
    $x^* = x + \delta_{y_a} + g$
  end while
  return x^*

5. 検証実験

5.1 データセット

本研究では，先行研究 [4] と検知率を比較するため，同一のデータセットを使用した．

データセットは，2019 年 1 月から 2020 年 3 月の間に HybridAnalysis から収集した PowerShell スクリプト 589 体，AnyRun から収集した PowerShell スクリプト 355 体，github から入手した良性の PowerShell スクリプト 5000 体からなる．HybridAnalysis からは，公開 API を用いて 1 日 4 回検体情報を検索し，公開されている PowerShell サンプルを入手した．AnyRun からは，拡張子に.ps が含まれるサンプルを検索し，手動でサンプルを入手した．github からは，まず PowerShell を含むすべてのリポジトリを検索し，上位 1000 件をダウンロードした．次に，拡張子に.ps が含まれているファイルを抽出し，これらをすべてサンプルとした．このように，複数の情報源から長期間でサンプルを収集していおり，サンプルにはある程度の網羅性が確保されているものと考えられる．悪性 PowerShell スクリプトは，一部のウイルス対策ソフトでは脅威として認識されていない．HybridAnalysis および AnyRun で収集したサンプルから，主要ベンダ 5 社 (Kaspersky，McAfee，Microsoft，Symantec および TrendMicro) のうち 2 社以上が脅威と判定したサンプルを抽出して悪性のラベルを付与した．悪性と判定したベンダが 0 社のサンプルには，良性のラベルを付与した．どちらにも該当しないサンプルは除外した．なお，github で収集したサンプルに対しても同様の調査を適用し，悪性または悪性疑いと判定されたサンプルは除外した．次に，データセットを時系列で分割した．理由としては，実際の悪性 PowerShell 検知を想定した場合，機械学習モデルの訓練に使用できるサンプルは既知のサンプルに限られるためである．HybridAnalysis および AnyRun から入手したサンプルは，投稿された日付に基づき，2019 年 6 月以前と 7 月以降で分割した．6 月以前のものを既知のサンプルとして訓練データおよび攻撃データに，7 月以降のものを未知のサンプルとして偽装データに使用した．github から入手したサンプルは，時系列情報が得られなかったためランダムに 2 分割し，一方を訓練データおよび攻撃データに，もう一方を偽装データに使用した．使用したデータセットの内訳は，表 1 のとおりである．

表 1 データセット内訳

AnyRun, HybridAnalysis			github
データセットの種別	悪性	良性	良性
訓練データ，攻撃データ	309	232	4901
偽装データ	171	92	

5.2 環境

実験に使用した環境については表 2 に，自然言語処理技術および分類器たる機械学習モデルの実装で使用したライブラリは表 3 に示すとおりである．

表 2 環境

CPU	Core i7-8700K 3.70GHz
Memory	16GB
OS	Windows10 Home
使用言語	Python2.7

表 3 使用した主なライブラリ

自然言語処理技術	Bag-of-Words Doc2Vec LSI	gensim -3.7.3
機械学習モデル	Bag-of-Words RandomForest	scikit-learn -0.20.4
	XGBoost	xgboost -0.82

5.3 実験内容

検証実験では，検証手法における偽装データへの挿入処理の回数を 0 から 500 回の範囲で変化させた．そして偽装データを各モデルに分類させ，悪性 PowerShell の再現率を測定した．この際，本実験で使用するデータセットは，すべて良性 PowerShell と悪性 PowerShell の数が不均衡なものとなっている．不均衡データに対して機械学習モデルの訓練を行うと，少数派に対する分類精度が著しく低下してしまう [19] ため，訓練データの良性 PowerShell に対してアンダーサンプリングを行った上で各実験を実施した．アンダーサンプリングは，訓練データに含まれる全ての良性 PowerShell から訓練データの悪性 PowerShell と同数をランダムに抽出する手法で行った．この処理ため，訓練データは試行ごとに異なるものを使用していることとなり，結果にばらつきが生じる．実験結果の精度を上げるため，測定においては試行を 10 回行い，その平均値を結果とした．なお，アンダーサンプリングは事前に良性 PowerShell と悪性 PowerShell の数の均衡が取れたデータセットを準備できる場合には不要である．そのため，本研究においてはアンダーサンプリングを検証手法ではなく実験内容の一部としている．また，検知モデルを実際に運用する場合においても，均衡の取れたデータセットを準備できると考えるため，実用的な精度の確保も満たしていると考える．偽装データの良性 PowerShell に対しては，アンダーサンプリングを適用しない．これは，実際の運用を想定した場合，悪性 PowerShell スクリプトと遭遇する確率は良性のものに対してかなり低いと考えられるためである．

5.4 実験結果

本節では，検証手法の効果を確認する．自然言語処理技術に BoW を用いた各モデルの結果を図 2 に，Doc2Vec を用いた各モデルの結果を図 3 に，LSI を用いた各モデルの結果を図 4 に示す．各グラフの縦軸は各モデルが分類した悪性 PowerShell の再現率の値，横軸は検証手法の挿入処理の回数である．本研究では，悪性 PowerShell の検知について，回避攻撃よる影響の程度を検証することから，再現率（Recall）に注目する．全ての自然言語処理技術と機械学習モデルの組み合わせの中で，500 回の挿入処理で最も再現率が低下したモデルは，Doc2Vec と XGBoost を組み合わせたモデルであった．その再現率は，挿入処理なしの場合から約 0.78 の低下であった．図 3 から，自然言語処理技術に Doc2Vec を用いた各モデルは，いずれも 10 回の挿入処理で再現率の大きな低下が確認できる．特に，RandomForest と組み合わせたモデルは 10 回の挿入処理で再現率が 0.77 から 0.39 へとほぼ半減した．図 4 から，自然言語処理技術に LSI を用いた各モデルは，組み合わせた機械学習モデルに関わらず，ほぼ同じ変化を示すことを確認した．図 2 から，Bag-of-Words を用いた分類器は，挿入処理による影響が最も少ないことを確認した．特に，XGBoost と組み合わせた分類器は 500 個の特徴を付与した場合の再現率は 0.76 であった．しかしながら，挿入処理なしの場合の再現率は 0.96 であり，約 21%の性能低下となった．

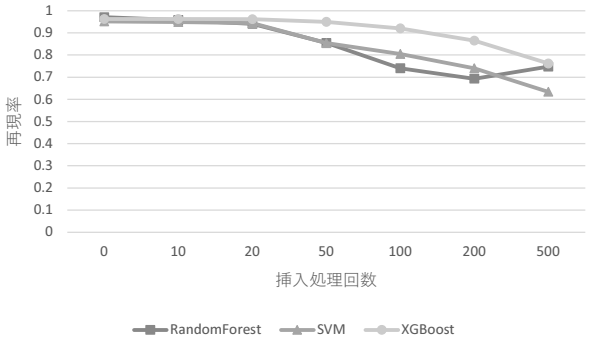

図 2 BoW を用いた各モデルの悪性 PowerShell 再現率の変化

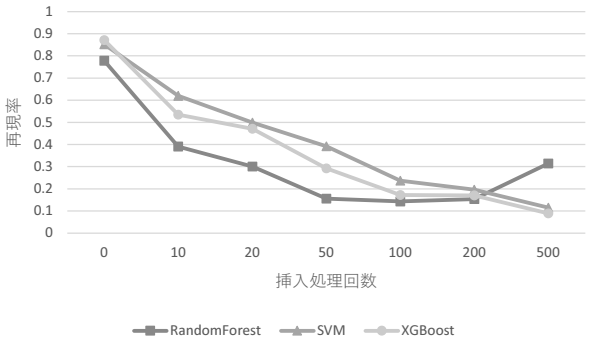

図 3 Doc2Vec を用いた各モデルの悪性 PowerShell 再現率の変化

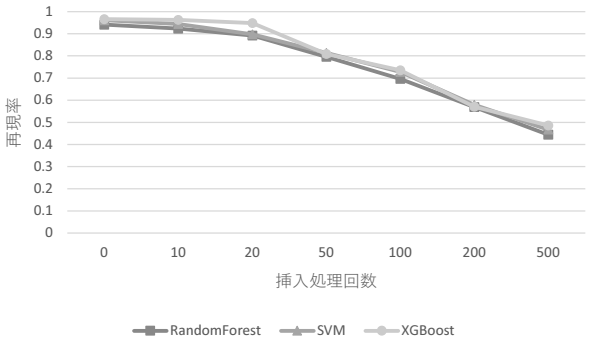

図 4 LSI を用いた各モデルの悪性 PowerShell 再現率の変化

6. 考察

6.1 機械学習モデルに対する回避攻撃の可能性

実験の結果，すべての自然言語処理技術と機械学習モデルの組み合わせで再現率の低下が確認された．自然言語処理技術に Doc2Vec および LSI も用いたモデルは，すべて最終的に再現率が 0.5 を下回っている．これは，半数以

上の悪性 PowerShell を見逃していることを意味し，実用的とは言い難い状態である．BoW を用いたモデルは再現率の低下が低く，XGBoost と組み合わせたモデルでは約 0.76 にとどまっている．しかしながら，これも 1/4 の悪性 PowerShell を見逃していることを意味している．また，挿入処理前は再現率が 0.96 であったことからも検知モデルとしては信頼性が十分でないと考える．さらに，図 2 のグラフ変化から，挿入処理をさらに追加した場合，一層の再現率低下も考えられる．したがって，Android マルウェアおよび VBA マルウエアの検知回避に有効であった本手法は，悪性 PowerShell の検知回避においても有効であると考えられる．よって，悪性 PowerShell 検知のための機械学習モデルに対する回避攻撃は発生し得ると考えられる．ただし，挿入処理に使用する単語および関数の種類が少ない場合，ソースコードに同じ単語が多数出現することになり，悪性 PowerShell の新たな特徴と見なされる可能性も考えられるため，注意が必要である．

なお，本実験で使用した攻撃データは偽装データよりも時系列的に古いものである．それにもかかわらず再現率の低下が確認されたという結果は，攻撃者が必ずしも最新のデータセットから良性 PowerShell スクリプトの要素を抽出する必要がないことを示していると考えられる．これは，良性 PowerShell スクリプトの性質に由来しているためと推察する．良性 PowerShell スクリプトは一般的にタスクの自動化や構成管理に用いられるため，マルウェアのような流行り廃りが見られにくく，特徴となる要素も時期による変化が小さいためと考える．

6.2 自然言語処理技術と機械学習モデル

本実験で得られた各グラフを比較すると，興味深い特徴が得られた．検証した各モデルは，用いる自然言語処理技術ごとに類似した再現率の変化を示している．特に，新たに回避手法を適用した Doc2Vec を用いたモデルはいずれも顕著に影響を受けている．これは，Doc2Vec の文書の特徴ベクトルを生成するという性質が影響していると推察する．Doc2Vec は，特徴ベクトルを生成するために文書と前後の単語に挟まれる単語を予測する．そこへ本実験では，良性 PowerShell スクリプトの特徴を新たに挿入したため，少ない挿入量でも文書全体の特徴ベクトルが大きく変化したものと考えられる．一方，LSI を用いたモデルは，機械学習モデルごとの差がほとんど認められない結果となった．これは，LSI が特異値分解を用いて文書と単語の関連性に重点を置いた特徴ベクトルを生成する性質が影響していると推察する．特異値分解によって表記ゆれや文書決定に貢献度の低い要素は排除され，重要な要素のみに集約される．これによって生成される特徴ベクトルは，いずれの機械学習モデルにとっても最適なものになっているため，機械学習モデルごとの差が認められなくなったと考えられる．以上のことから，検証手法による影響は自然言語処理技術に依存していることが考えられる．

6.3 研究倫理

先行研究 [2], [10], [15] では大規模なデータセットを使用しているが，これはセキュリティベンダ等が独自収集したものであり，一般的に提供を受けることは困難である．本研究では，PowerShell のサンプルは全て公開されているものを使用している．また，自然言語処理技術および機械学習モデルの実装に使用した gensim および scikit-learn 等のライブラリも無償公開されている．したがって，本研究と同様の環境構築は容易であり，本研究の再現性は高いと考える．

6.4 研究の限界

本研究では，先行研究 [4] と同等数のサンプルを用いて実験を行ったものの，悪性 PowerShell のサンプル数は必ずしも十分であるとは言い難い．これは，再現性を確保するため，任意に利用可能なマルウェア等の配布サイトから悪性 PowerShell のサンプルを収集したためである．実験に用いるサンプル数を増やすことができれば，検証の精度を向上させることができると考える．また本研究は，回避攻撃の際に悪性 PowerShell スクリプトとしての動作を維持させることにも着意して実験を行った．理論的には悪性 PowerShell スクリプトの動作を維持していると考えるが，確認のためのサンドボックスを用意できなかったため，実際に PowerShell スクリプトを実行しての確認はできていない．

7. おわりに

本研究では，悪性 PowerShell スクリプトに対して，良性サンプルの特徴を付与することにより，その機能を維持しつつ，機械学習モデルに対する回避攻撃が可能であることを示した．また，新たに Doc2Vec に対しても回避手法を適用するとともに，RandomForest および XGBoost に対してもサンプルを分類させ，その効果を検証した．本研究は，これらに対する回避手法の有効性を検証した初めての研究であるため，機械学習モデルに対する回避攻撃に関して新たな知見を提供することができた．検証実験の結果，Doc2Vec を用いたモデルは他のモデルよりも回避攻撃の影響を受けやすいことを確認した．また，回避攻撃による影響は自然言語処理技術に依存しており，LSI では機械学習モデルごとの差がほとんど認められないことを確認した．

今後の課題としては，他の自然言語処理技術および機械学習モデルを適用した検証が挙げられる．双方とも多くの手法が考案されており，今回検証した自然言語処理技術と機械学習モデルの組み合わせはごく一部である．他の自然言語処理技術と機械学習モデルを組み合わせることによ

り，今回検証した回避攻撃に耐性を有するものが発見できる可能性も考えられる．

謝辞 本研究は JSPS 科研費 21K11898 の助成を受けたものです．

参考文献

[1] WatchGuard Technologies: Internet Security Report - Q4 2020, `https://www.watchguard.com/wgrd-resource-center/security-report-q4-2020` (2021). (Accessed on 07/21/2021).

[2] Hendler, D., Kels, S. and Rubin, A.: Detecting Malicious PowerShell Commands using Deep Neural Networks, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018* (Kim, J., Ahn, G., Kim, S., Kim, Y., López, J. and Kim, T., eds.), ACM, pp. 187–197 (online), DOI: 10.1145/3196494.3196511 (2018).

[3] Tajiri, Y. and Mimura, M.: Detection of Malicious PowerShell Using Word-Level Language Models, *Advances in Information and Computer Security - 15th International Workshop on Security, IWSEC 2020, Fukui, Japan, September 2-4, 2020, Proceedings* (Aoki, K. and Kanaoka, A., eds.), Lecture Notes in Computer Science, Vol. 12231, Springer, pp. 39–56 (online), DOI: 10.1007/978-3-030-58208-1_3 (2020).

[4] 田尻裕貴，三村　守：単語ベースの機械学習モデルによる未知の悪性 PowerShell スクリプトの検知手法，情報処理学会論文誌，Vol. 62, No. 5, pp. 1317–1327 (2021).

[5] Chen, S., Xue, M., Fan, L., Hao, S., Xu, L., Zhu, H. and Li, B.: Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach, *Comput. Secur.*, Vol. 73, pp. 326–344 (online), DOI: 10.1016/j.cose.2017.11.007 (2018).

[6] Maiorca, D., Biggio, B. and Giacinto, G.: Towards Adversarial Malware Detection: Lessons Learned from PDF-based Attacks, *ACM Comput. Surv.*, Vol. 52, No. 4, pp. 78:1–78:36 (online), DOI: 10.1145/3332184 (2019).

[7] Chen, B., Ren, Z., Yu, C., Hussain, I. and Liu, J.: Adversarial Examples for CNN-Based Malware Detectors, *IEEE Access*, Vol. 7, pp. 54360–54371 (online), DOI: 10.1109/ACCESS.2019.2913439 (2019).

[8] Grosse, K., Papernot, N., Manoharan, P., Backes, M. and McDaniel, P. D.: Adversarial Examples for Malware Detection, *Computer Security - ESORICS 2017 - 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II* (Foley, S. N., Gollmann, D. and Snekkenes, E., eds.), Lecture Notes in Computer Science, Vol. 10493, Springer, pp. 62–79 (online), DOI: 10.1007/978-3-319-66399-9_4 (2017).

[9] 山本理紗，三村　守：機械学習を悪用する未知のマクロマルウェアの脅威，技術報告 51，防衛大学校情報工学科，防衛大学校情報工学科 (2020).

[10] Ugarte, D., Maiorca, D., Cara, F. and Giacinto, G.: PowerDrive: Accurate De-obfuscation and Analysis of PowerShell Malware, *Detection of Intrusions and Malware, and Vulnerability Assessment - 16th International Conference, DIMVA 2019, Gothenburg, Sweden, June 19-20, 2019, Proceedings* (Perdisci, R., Maurice, C., Giacinto, G. and Almgren, M., eds.), Lecture Notes in Computer Science, Vol. 11543, Springer, pp. 240–259 (online), DOI: 10.1007/978-3-030-22038-9_12 (2019).

[11] White, J.: Practical Behavioral Profiling of PowerShell Scripts through Static Analysis (Part 1), `https://unit42.paloaltonetworks.com/practical-behavioral-profiling-of-powershell-scripts-through-static-analysis-part-1/`. (Accessed on 08/20/2021).

[12] Liu, C., Xia, B., Yu, M. and Liu, Y.: PSDEM: A Feasible De-Obfuscation Method for Malicious PowerShell Detection, *2018 IEEE Symposium on Computers and Communications, ISCC 2018, Natal, Brazil, June 25-28, 2018*, IEEE, pp. 825–831 (online), DOI: 10.1109/ISCC.2018.8538691 (2018).

[13] Li, Z., Chen, Q. A., Xiong, C., Chen, Y., Zhu, T. and Yang, H.: Effective and Light-Weight Deobfuscation and Semantic-Aware Attack Detection for PowerShell Scripts, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019* (Cavallaro, L., Kinder, J., Wang, X. and Katz, J., eds.), ACM, pp. 1831–1847 (online), DOI: 10.1145/3319535.3363187 (2019).

[14] Hendler, D., Kels, S. and Rubin, A.: AMSI-Based Detection of Malicious PowerShell Code Using Contextual Embeddings, *ASIA CCS '20: The 15th ACM Asia Conference on Computer and Communications Security, Taipei, Taiwan, October 5-9, 2020* (Sun, H., Shieh, S., Gu, G. and Ateniese, G., eds.), ACM, pp. 679–693 (online), DOI: 10.1145/3320269.3384742 (2020).

[15] Rusak, G., Al-Dujaili, A. and O'Reilly, U.: AST-Based Deep Learning for Detecting Malicious PowerShell, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018* (Lie, D., Mannan, M., Backes, M. and Wang, X., eds.), ACM, pp. 2276–2278 (online), DOI: 10.1145/3243734.3278496 (2018).

[16] White, J.: Pulling Back the Curtains on EncodedCommand PowerShell Attacks, `https://unit42.paloaltonetworks.com/unit42-pulling-back-the-curtains-on-encodedcommand-powershell-attacks/`. (Accessed on 08/20/2021).

[17] Fang, Y., Zhou, X. and Huang, C.: Effective method for detecting malicious PowerShell scripts based on hybrid features, *Neurocomputing*, Vol. 448, pp. 30–39 (online), DOI: 10.1016/j.neucom.2021.03.117 (2021).

[18] Le, Q. V. and Mikolov, T.: Distributed Representations of Sentences and Documents, *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, JMLR Workshop and Conference Proceedings, Vol. 32, JMLR.org, pp. 1188–1196 (online), available from ⟨http://proceedings.mlr.press/v32/le14.html⟩ (2014).

[19] Japkowicz, N.: The Class Imbalance Problem: Significance and Strategies, *In Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI*, pp. 111–117 (2000).