類似性に基づくハニーポット収集データの分類

小関 純 [1,] 佐藤 大造 [1,] 阿曽村 一郎 [2,]

概要：ハニーポットで収集されるファイルを解析することは，多種多様な攻撃に利用される未知のマルウェアを調査するために有効である．その一方で，収集されるデータの中には攻撃とは無関係のファイルや，内容の酷似するものも大量に存在する．これらハニーポットに設置されたファイルをマルウェア調査に活用するためには，収集データの中から悪性ファイルを効率的に検知する必要がある．本稿では，ハニーポットに設置されたファイルから悪性ファイルを機械学習によって分類するシステムを提案し，解析処理の効率化を図る検証を行う．攻撃者により設置されたファイルから特徴を抽出したのち，悪性ファイルを学習する分類器を作成し予測結果から得られた知見について述べる．

キーワード：マルウェア，ハニーポット，シェルスクリプト，機械学習，自然言語処理

Classification based on structural similarities of data collected from honeypots

Jun Koseki[1,] Daizo Sato[1,] Ichiro Asomura[2,]

Abstract: Honeypot is effective system to collect unknown malwares used for various attacks. However, honeypots collect large amount of malwares and these subspecies. They collect not only malicious files but also benign files. In order to do malware analysis using collected files from honeypots, effective and efficient classification in malicious files from large number of unknown files is required. In this paper, we propose machine learning based malicious file classification system and evaluate possibility of improving efficiency in malware analysis. After feature extraction from collected files using honey pots, we created malicious file classifier. We described the findings based on a predicted result using the classifier.

Keywords: Malware, Honey pot, Shell script, Machine learning, Natural language processing

1. はじめに

近年, IoT システムに関する研究開発と社会実装が急速に展開されていく中で，IoT 機器に関するセキュリティ対策が問題となっている. 2016 年に猛威を振るった Mirai は，多様な亜種を登場させつつ依然としてその勢力を維持し続けている[1]． また，Linux サーバや IoT デバイスを標的とした Perl 製 IRC bot による，仮想通貨の発掘や DDoS 攻撃を試みる活動も，新たな攻撃手法が次々と観測されている[2]．これら IoT 機器を標的としたサイバー攻撃は，不正な SSH 接続を通じたデバイスへの侵入が原因で発生することが多い．SSH 自体は，通信経路が暗号化されるため安全なプロトコルとされるが，パスワードの流出や総当たり攻撃などによって侵入され，攻撃者によるコマンドの実行やシェルスクリプトの設置などにより，前述のような被害を受ける場合がある．また，改変されたコードの販売や，第三者のリークによる参入障壁の低下，各 IoT 機器固有の脆弱性を利用した活動などにより，その攻撃手法にも多様化が見られている[3]．ハニーポットで観測されるデータを活用し新たに発生していく攻撃の実態調査を行うことは，これら多様化するサイバー攻撃を早期発見するために有効である．

OpenSSH を模した低対話型ハニーポット Cowrie は，侵入者によって設置されたファイルが download というディレクトリ内に格納される[4]．格納されたファイルには，マルウェアのダウンローダとして機能するシェルスクリプトや，C2 サーバとの通信を試みる bot 等を多数確認することができる．これらの収集データを分析することにより，侵入者の攻撃手法や通信先等，未知のマルウェアを調査する際に有効となるデータを収集することができる．一方で，観測されるファイルの中には，攻撃とは無関係なスクリプト等，解析対象として意味を持たないファイルも数多く存在する．さらに，類似した行動パターンで同一のマルウェアをダウンロードさせるプログラムも多数含まれるため，これら雑多なファイルを含む膨大な収集データをマルウェア調査に利用するには，多大な時間を要する．

本稿では，収集されたデータのうち悪性ファイルを効率

1 株式会社ラック
LAC
2 みずほフィナンシャルグループ
Mizuho Financial Group, Inc,

よく解析するため，Cowrie に設置されるファイルを可読文字列の類似性に基づいて分類する手法を提案する．分類器によって検知されたファイルのみを解析対象とすることにより，少量のファイル調査結果から観測データの全体像を把握することを可能とし，ハニーポットの解析コストを削減させることを目標とする．

2. 関連研究・技術

マルウェアの識別および分類に機械学習を応用した研究においては，これまでにも，多様な視点で既知マルウェアとの類似度を比較する検証が行われている．本章では，学習アルゴリズムおよび分類対象データのアプローチから，関連する技術について述べる．

田中らは，Windows の実行ファイル中に含まれる PE ヘッダ情報に着目して検証を行っている[5]．特徴量として，取得したヘッダ情報から絞り込んだ 10 種類の変数を利用し，ロジスティック回帰分析によってモデルの構築を行っている．ロジスティック回帰は機械学習における分類モデルの一つであり，線形分離可能なクラスに対して高い性能を発揮する．また，田中らの検証においては学習モデルにロジスティック回帰を使用することにより，変数ごとのオッズ比を求めることができる．これにより，選定された変数の妥当性を評価することが可能となる．

動的解析の結果に含まれる API コール群を用いてマルウェア亜種の判定を行う佐藤らの検証では，API 名と引数の列を一つの文章として扱い, Word Vector および Word Vector を拡張した Paragraph Vector を用いて表現する手法を提案している [6-7]． Word Vector とは，文書中に出現する単語の特徴を，その前後に出現する単語をもとにベクトル化する手法であり，その応用である Paragraph Vector は，Word Vector で得られる特徴を持つベクトル表現を，文書についても得ることができるよう拡張したものである[8].

上述の研究は，実行ファイルの解析結果から抽出された文字列をもとに分類を試みる検証である．一方で Dumont らは，不正な SSH 接続を行う攻撃者の挙動を，実行されたシェルコマンドに基づいて検知する手法を提案している[9]．N-gram によるアプローチからコマンドをベクトル化して特徴量を抽出し，K 近傍方によるクラス分類を検証している．SSH を通じて端末への不正ログインを行うシェルセッションから攻撃者を識別し，一般の利用者と不正にアクセスを試みる侵入者とを分類する検証であり，静的解析されたプログラムコードや命令列を必要とせず，シェルコマンドから悪意ある挙動を分類するという観点において，本稿のアプローチと近似する研究である．

Dumont らの検証では, 良性ユーザと悪性ユーザのコマンドデータを次のように定義，収集している．攻撃者が自身のファイルをインターネット上に公開する可能性は低いと考え，GitHub 上に公開されている Bash シェルのヒストリログファイルを，一般の開発者によって公開された良性コマンドであると分類．対して，Cowrie などハニーポットで収集された実行コマンド等については，無害なユーザであれば未知のデバイスであるハニーポットへ侵入することはないとし，観測されたシェルセッションはすべて悪性ユーザによるものと定義している．

3. 提案

前述の Dumont らの研究[9]では，シェルコマンドから単語 N-gram を抽出し，良性ユーザ・悪性ユーザを分類する検証を行う．入力されたコマンドから悪意ある挙動を検知するという観点において，本稿と関連する研究であるといえる．しかし，デバイスへの不正アクセス自体を悪意ある挙動ととらえているため，ハニーポットで収集されるデータについてはすべて悪性と定義している. 本稿においては，ハニーポットにおけるファイル解析の効率化を図る検証であるため, 侵入者によって Cowrie に設置されたファイルおよび記述されるコマンド群から，何らかの破壊活動に利用される悪性ファイルを分類する必要がある．

本章では分類器の構築にあたり，悪性ファイルの特徴量生成を二つの観点から検証する．ここでは運用中の Cowrie に設置されるファイルのうち，悪性ファイル・良性ファイルを以下のように定義する．観測されるファイルには，何らかの設定ファイルの断片コード，デバイスの通信速度を測定するのみのスクリプト等，実際の攻撃と直接的な関係のないファイルが多く観測されるが，これらはすべて良性ファイルとして学習データを作成した．

悪性ファイル(malicious)

1. Unix 系 OS で実行可能なシェルスクリプト
2. マルウェアのダウンロード元ドメインからファイルをダウンロードするプログラムと思われる記述がされている．

良性ファイル(benign)

1. 直接的な攻撃プログラムでないファイル
2. 悪性ファイルと判定されたもの以外全般

3.1 提案手法 1

3.1.1 重みづけ評価

運用中の Cowrie に多数観測される, 悪性ファイルのプログラムコードに着目して特徴量の抽出を行う．プログラムコードに見られる悪性ファイルとしての挙動を，その重要度と出現頻度に応じて点数を与える．ファイルの悪性度を実行コマンド，通信先の二値データに分割して評価し，二次元空間に配置した．これにより，分類結果の直感的な理

解を可能とし，検体間の関係性を可視化表現することができる．良性ファイルを 0，悪性ファイルを 1 としたラベルを付与し，ファイルごとのデータセットを作成する．

表 1　得点表(実行コマンド別)

wget コマンドを使用	10 点
chmod コマンドを使用	5 点
curl コマンドを使用	3 点
コピーしたファイルの削除を試みる	3 点
nohup コマンドを使用	3 点
busybox の指定	10 点

表 2　得点表(通信先別)

通信先が IP で記述されている	10 点
通信先 TLD が.ch	2 点
通信先 TLD が.ru	2 点
通信先 TLD が.tk	3 点
通信先 TLD が.jp 以外	3 点
通信先フォルダ名が tmp	3 点
通信先ファイル拡張子が.exe	5 点

得点表をもとに作成した分布図(図 1)から，悪性・良性ファイルの境界線となる直線を求めるため，ロジスティック回帰分析による学習モデルの構築を行う．前述のようにロジスティック回帰モデルは，線形分離可能なクラスに対して二値分類タスクを行う場合において，高い性能を発揮する学習モデルである．モデルの構築には，機械学習用 Python ライブラリ scikit-learn[10]を利用した．

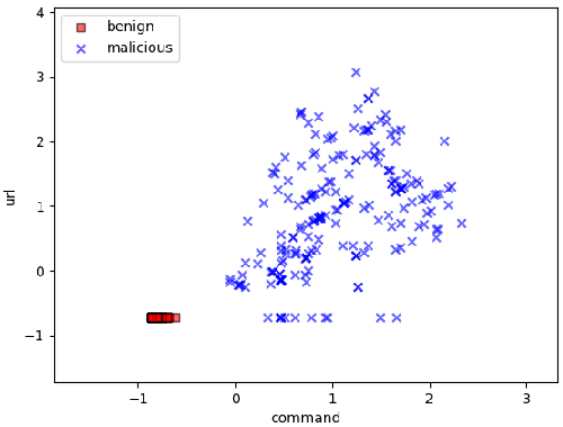

図 1　正規化した得点から生成した分布図

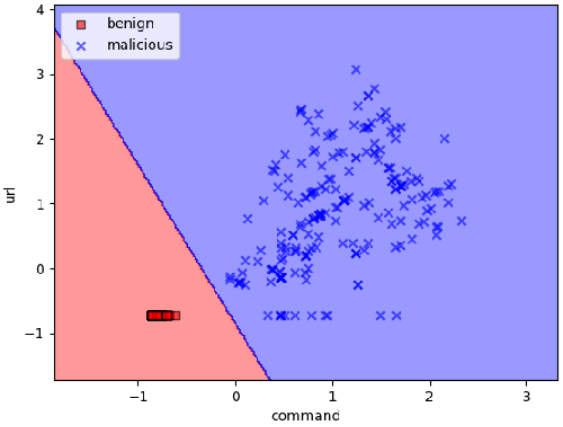

図 2　訓練結果をもとに境界線をプロット

3.1.2 重みづけ評価の検証

経験則に基づき，コマンドおよび通信先の出現頻度に沿って評価得点の設定を行ったが，実際のデータとの剥離が懸念される．そのため，実際のファイルを手動で解析した結果と比較し，その妥当性を検討した．検証用ファイルとして，Cowrie から取得した 6,378 個のうち，同一ファイルの重複を除いた悪性ファイル 268 個を抽出した．

抽出したファイルの内容から，1 ファイル当たりのコマンドの出現回数を計測した．結果は以下の通り．

表 3　1 ファイル当たりのコマンドの出現回数

コマンド	回数	備考
rm	5.874524715	ファイル削除
chmod	3.680608365	権限変更
wget	2.855513308	ファイル取得
curl	1.269961977	ファイル取得
kill	1.193916350	プロセスキル
ftpget	1.015209125	ファイル取得
pkill	0.711026616	プロセスキル
sleep	0.686567164	プロセス待機
nohup	0.642585551	ハングアップシグナル無効
killall	0.463878327	プロセスキル
busybox	0.015209125	busybox 指定

前述 3.1 提案方法 1 の重みづけと比較すると、頻度順が前後する箇所がいくつか見られるが、項目自体は概ね一致する。頻度としては rm コマンドの重要性は高いが攻撃者が準備するファイル数によって大きくばらつきがあるため，3 点の得点は妥当と判断する．一方でマルウェア配置を行うための「挙動」としての観点から，悪性ドメインからのダウンロードの起点となる wget や権限変更の chmod の方を重視し得点をそれぞれ 10 点，5 点と高めに設定したことも妥当であると判断する．busybox の指定は，出現頻度自体は高くないが，IoT 機器に対する攻撃の起点に利用されることが多いため，得点を 10 点とした．

次に，1 ファイル当たりの通信先の出現回数を計測した．前述の得点表は，通信先の指定が IP アドレスによる場合は悪性度が高いという仮説のもと作成している．実際に検体の調査を行ったところ，記述されている通信先は IP アドレス指定のものが大部分を占めていたため，IP アドレスの記述を重視した表 2 の得点ルールは妥当であると考えることができる(表 4).

表 4　1 ファイル当たりの通信先の出現回数

通信先	回数
通信先が IP で記述されている	5.026119402
通信先 TLD が.tk	0.011194029
通信先 TLD が.eu	0.003731343
通信先 TLD が.net	0.003731343
通信先 TLD が.club	0.011194029
通信先 TLD が.org	0.003731343
通信先 TLD が.com	0.022388059
通信先フォルダ名が tmp	0.000000000
通信先ファイル拡張子が.exe	0.000000000

3.2 提案手法２

3.1 では，設置されたファイルに含まれる単語の重要度を過去の観測結果に基づいて選定し，良性・悪性の二項に分類した．しかし，選定した単語の重要度は経験則をもとに手動で導き出したものであるため，未知の特徴量抽出が困難である．

そこで，収集された全ファイルの単語群から，各ファイルにおける単語の出現頻度を算出し，Bag of Words(以下 BoW)によるベクトル表現を利用して特徴量を抽出する．BoW は，ある集合に存在する単語が文章中にいくつ出現するかを特徴とする，出現順序等を考慮しないシンプルなベクトル表現手法である．

Cowrie にて収集された全ファイルの単語集合からコーパスを構築し，BoW によるベクトル表現を生成する．例えば，全ファイル中の単語が表 5 の 5 語のみで構成されている場合，”#!/bin/sh　killall　nohup”と記述された文章は，((1, 1),(2, 0),(3, 1),(4, 0), (5, 1))と表現することができる．コーパスの構築および BoW の生成は，トピックモデルおよび自然言語処理用 Python ライブラリの gensim[11]を利用した．また，単語コーパスの構築にあたり，半角スペースあるいはカンマで分割された，特定の記号を除外した文字列を一つの単語と定義した．

表 5　単語コーパス(例)

ID	単語	出現回数
1	#!/bin/bash	69
2	#!/bin/sh	251
3	killall	46
4	rsync*	82
5	nohup	81

構築したコーパスをもとに生成した BoW は，単語の出現頻度のみを表現したベクトルであるため，単語の重要度が考慮されていない．例えば，出現回数の多い”#!/bin/bash”や”#!/bin/sh”は，UNIX/Linux OS において，実行するインタープリタを示すために記述されるシバンラインと呼ばれる構文であり，悪性・良性問わずシェルスクリプトの先頭行に頻出する．対して”wget”や”chmod”などのコマンドは，3.1 提案手法 1 で述べたように，マルウェア配置を行う悪性ファイルの挙動として重視すべきコマンドである．これら単語の重要度を表す指標として，TF-IDF(Term Frequency-Invers Document Frequency)による重み付けを行う．

TF-IDF とは，TF(Term Frequency: 単語の出現頻度)と IDF(Inverse Document Frequency: 逆文書頻度)の積として定義される重み付けの指標である．

$$tfidf(t,d) = tf(t,d) \times idf(t,d)$$

ここで逆文書頻度idf(t, d)は，以下のように定義される．

$$idf(t,d) = log\frac{n_d}{1+df(t,d)}$$

n_dはファイルの総数を示す．$df(t,d)$は単語tを含む文書dの総数を示し，ゼロ除算を回避するため分母に 1 を加算する．

gensim のモジュールで TF-IDF による重み付けを行った特徴ベクトルをもとに，分類器の構築を行う．学習モデルの生成には，scikit-learn のモジュールを利用した．分類アルゴリズムとしてロジスティック回帰，ランダムフォレスト，サポートベクターマシンを選択し，それぞれの正答率を検証した．検証の結果，比較的安定した正答率を算出するロジスティック回帰を採用し，分類器の構築を行った．

4. 評価実験

本章では，3 章で提案した二つの特徴量抽出手法に対して，実際に未知ファイルの予測を行う API を作成し，その精度を評価する．

4.1 実験環境

評価実験を行うにあたって，T-Pot を東京，シドニー，ロンドン，アムステルダム，フランクフルト，バンガロール，シンガポール，サンフランシスコ，トロント，ニューヨーク，ダラスの 11 地域に設置し，4 か月観測を続けた．観測データの中から Cowrie にて設置された約 300 件のファイルを採取，8 割を学習データとして，2 割のファイルを検証用のテストデータとして利用した．

T-Pot は，用途の異なる数種類のハニーポットを同一ホスト上で運用するためのプラットフォームであり，Cowrie を含む複数種類のハニーポットがそれぞれ独立した Docker コンテナとしてサポートされている[12].

収集データをもとに分類器を構築し，記述される文字列

をもとに悪性・良性を分類する API を Python で作成，その性能を評価した．

4.2 評価指標

分類精度の評価指標には混同行列を採用した．悪性ファイルを Positive，良性ファイルを Negative とし，予測されたクラスと実際のクラスの組み合わせにより，真陽性(TP: True Positive)，真陰性(TN: True Negative)，偽陽性(FP: False Positive)，偽陰性(FN: False Negative)の 4 種類に分類する．真陽性は悪性と判定された悪性ファイル，真陰性は良性と判定された良性ファイル，偽陽性は悪性と判定された良性ファイル，偽陰性は良性と判定された悪性ファイルをそれぞれ示す．

表 6　混合行列

	悪性判定	良性判定
悪性ファイル	真陽性(TP)	偽陰性(FN)
良性ファイル	偽陽性(FP)	真陰性(TN)

これらの混合行列から算出される， Recall(再現率)，Precision(適合率)，F-measure(F 値)を性能指標として評価する．Recall は，悪性と予測したファイルに含まれる悪性ファイルの割合を示すため，良性判定された悪性ファイルの数を示す指標となる．Precision は，全悪性ファイルのうち悪性と予測することができたファイルの割合であり，誤検知された良性ファイルの数を示す指標となる. F-measure は，再現性と適合率の調和平均をとった網羅性を表す指標となる．3 章で提案した二つの特徴量抽出手法をもとに分類器を作成し，それぞれの分類精度を算出した(表 7).

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F - measure = \frac{2Recall + Precision}{Recall + Precision}$$

表 7　実験結果

	Recall	Precision	F-measure
提案手法 1	0.913	0.9	0.98
提案手法 2	0.968	0.984	0.976

4.3 性能評価

本稿は, 悪性ファイルの分類システム構築による, Cowrie を活用したマルウェア調査の効率化を目標としている．設置されるデータからマルウェア配置に利用されるシェルスクリプト等を事前に検知，分類することにより，調査対象ファイルの総数を削減することが可能となる．システムによって良性と判定されたファイルは，調査対象となりうる可能性が低いファイルと判断されるため, 偽陰性(良性判定された悪性ファイル)の少ないモデルであることが望ましい。前述の通り，Recall(再現率)は偽陰性を表す指標ととらえることができるため，性能評価をする際に重要視する必要がある．

提案 1，2 いずれの特徴量抽出手法を採用した場合においても，各指標は 9 割以上の数値を示している．しかし，限られた学習データをもとに分類を行っているため，過学習の可能性を考慮する必要がある. 今後も Cowrie の観測を継続していく中で，蓄積される学習データの法則性を調査し，学習モデルの再検討を行う必要がある．

5. 考察

本章では，分類器の性能評価を踏まえ，今後の課題，システムとしての有用性について考察する．

5.1 課題

本稿では, 運用中の Cowrie にて収集された悪性ファイルの分類を, 記述される文字列の類似性に基づいて検証した．観測データに頻出する，マルウェア配置を行うシェルスクリプトに焦点を当ててファイルを分類した結果，検証では 9 割以上の悪性ファイルを分類することができた．実験結果の数値では高い精度を示したが，前述のように本検証の実験結果は，特定の運用期間内に収集されたデータのみに依存したものであるため，学習データの不足による過学習の発生を考慮すべきである．

3.1 提案手法 1 では，悪性ファイルに頻出するコマンドおよび通信先をもとに得点ルールを作成，特徴量として利用した．評価実験においては，手法 2 よりも高い適合率を示している．分類システムとして運用するためには，継続的なデータの蓄積に加え，悪性ファイルの傾向分析等を行いつつ，適宜，得点基準の見直しを行う必要がある．

一方, 3.2 提案方法 2 では, 単語の出現頻度をもとに BoW によるベクトル表現を生成する手法で特徴量を抽出し，良性/悪性の二項に分類した．BoW は固定長のベクトル表現を取得する最もシンプルかつポピュラーな手法ではあるが, 順序情報，単語間のセマンティクスの喪失という欠点がある．文脈の特徴からファイルを分類する場合，特徴量の抽出手法を工夫しなければならない．

例えば，Mirai 系マルウェアのダウンローダ等においては，記述されるプログラムコードから，頻出する一連の流れを確認することができる．出現単語だけでなく，プログラムコードの一連の流れから悪性ファイルを検出する場合, 学習データの更なる収集に加え，プログラムコード間の距離尺度に基づいた分類手法を採用する必要がある。

5.2 ツールとしての有用性

観測期間中の悪性ファイルにおいては，記述内容の類似するプログラムが多数存在したため，極めてシンプルな学

習アプローチから 9 割以上の悪性ファイルを分類することに成功した.

調査対象となりうるファイルをあらかじめ検知・分類することにより，疑わしいファイルを優先的に調査することができるようになり, 従来, 6 日程度の工数を要した Cowrie 収集データにおける解析業務を，約 5 分程度にまで短縮することができた.

今後，抽出すべき特徴や学習アルゴリズムの再検討，悪性ファイルの種別を複数クラスに分類するための拡張など, 分類技術の洗練化を進める必要がある．これらの課題に当たり, 観測データの収集･分析を継続していく必要がある.

参考

[1] “IIJ-SECT 2018 年の IoT ボット観測状況と最近の動向“. https://sect.iij.ad.jp/d/2019/01/288147.html#fnref:footnote20

[2] “TREND MICRO セキュリティブログ [ハッキング集団「Outlaw」のボットネットを確認、仮想通貨発掘ツールと Perl ベースのバックドアを拡散]” https://blog.trendmicro.co.jp/archives/21709

[3] 国立研究開発法人 情報通信研究機構 サイバーセキュリティ研究所 サイバーセキュリティ研究室：”NICTER 観測レポート 2018” https://www.nict.go.jp/cyber/report/NICTER_report_2018.pdf

[4] https://github.com/cowrie/cowrie

[5] 田中 恭之，後藤 厚宏：統計的方法を用いた未知マルウェア検出手法の提案と評価，情報処理学会論文誌,2016,Vol.57

[6] 佐藤 拓未，後藤 滋樹：Word Vector を用いた亜種マルウェア判別法，早稲田大学修士論文, 5-35(2017-01-30)

[7] 佐藤 拓未，後藤 滋樹：Paragraph Vector を用いたマルウェアの亜種推定法, Computer Security Symposium 2016, 11 – 13 October 2016

[8] Quoc Le, Tomas Mikolov, “Distributed Representations of Sentences and Documents” in 2014, *Proceedings of The 31st International Conference on Machine Learning (ICML 2014), pp. 1188 – 1196*

[9] Pierre Dumont, Roland Meier, David Gugelmann, Vincent, “Lenders Detection of Malicious Remote Shell Sessions”, in *2019 11th International Conference on Cyber Conflict*

[10] https://scikit-learn.org/stable/

[11] https://pypi.org/project/gensim/

[12] https://github.com/dtag-dev-sec/tpotce