卒業論文　2018年度（平成30年度）

マルウェア解析に向けて
判断基準を提示する機械学習モデル

慶應義塾大学 法学部法律学科

野間口圭

徳田・村井・楠本・中村・高汐・バンミーター・植原・三次・中澤・
武田 合同研究プロジェクト

2018年1月

卒業論文　2018 年度（平成 30 年度）

マルウェア解析に向けて
判断基準を提示する機械学習モデル

論文要旨

マルウェアによってインターネットのセキュリティが脅かされている．マルウェアによる攻撃に対して対策を施すため，様々な研究や製品で教師あり機械学習を用いた手法が提案されている，これらの研究，製品は，人力による詳細な解析なしに学習結果を判定できる一方，機械学習モデル自体の判断基準が不明確である．従って当該製品，モデルの信頼性を担保することができず，また機械学習自体に対する攻撃によって容易に撹乱されるという問題がある．

この不明確さに対処する為，マルウェア解析の為に新しい機械学習モデルを提案する．提案されるモデルでは，モデルの検知，分類結果だけではなく，結果に影響を与えた検体の具体的な部分を提示する．これを実現する為，近似アルゴリズムである LIME を用いる．実験として，ファイルへのアクセスログを元にランサムウェアのファミリを分類するタスクを用いて評価する．

本手法により機械学習を使用したマルウェア分類モデルに意味解釈性を与えることで，マルウェアの被害を根絶した社会の実現に近づくことが期待される．

キーワード

機械学習，マルウェア解析，データ解析

慶應義塾大学 法学部法律学科

野間口圭

Abstract Of Bachelor Thesis Academic Year 2018

Machine Learning models indicating concrete criteria for malware analysis

Summary

Nowadays, malware has been threatening Internet security. To detect and classify malware, a lot of research and products using machine learning have been proposed, especially for end point detection. Whilst the research and products are arguing that they have high accuracy in various constraints, it is insufficient to give the models' result from the perspective of malware analysts, for they want clues of results for further investigation. Especially in case of using complex models, it is much more difficult to know the clues.

In this thesis we will present new machine learning models for malware analysis. the models will indicate not only detection result, but also concrete part of feature that are attributed to detection or classification result. To realize that, we will use algorithm called LIME to approximate complex machine learning models. As an examination, we use this models for ransomware classification and make sure that the extracted features are appropriate.

By using this models, it is expected that malware analysts can take countermeasure for malware by getting the clues of classification result.

Keywords

Machine learning, Malware analysis, Data analysis

Faculty of Law
Keio University

Kei Nomaguchi

目 次

図 目 次

表 目 次

第1章　序論

ここでは，マルウェアの現況と，これに対してどのような対策がとられてきたのか詳述する．

1.1　背景

1.1.1　マルウェア

インターネットは最早人々の生活にはなくてはならないものになっている．しかし，インターネットのセキュリティを脅かす脅威が存在する．その脅威の一つとしてマルウェアが挙げられる．マルウェアとは，財の窃取など，悪意を実現する機能を持ったプログラムの総称である．2018 年現在，それらが及ぼす被害は甚大であることが多い．例えばファイルの削除，入力情報の窃取，暗号化されたファイルを復号するための身代金要求などである．以下が，一般的に言われている主なマルウェアの種類を示した表である．最近では，これらの特徴を複数兼ね備えたものが多い 1.1.

表 1.1: マルウェアの種類

種類	説明
バックドア	インストールすることで攻撃者側からのアクセスや実行を可能にするもの.
ボットネット	バックドアと同じだが，感染端末は一人の攻撃者の指令に一律に動作する.
ダウンローダー	他の C2 サーバーから別種のマルウェアをインストールするもの.
キーロガー	キーストロークの内容を攻撃者側に伝える.
ルートキット	OS の脆弱性を利用して，マルウェアの存在や行動を隠すもの.
スケアウェア	虚偽の通知を画面に提示することで金銭や情報の窃取をするもの.
ワーム	他の端末や端末内で自己増殖するもの
トロイの木馬	正常なソフトウェアを装って不正な機能を実行するもの.
ランサムウェア	ファイルやリソースのアクセスを制限し，その解除に金銭を要求するもの.

1.1.2 本研究の目的

研究では，これに対応するため，機械学習技術を用いた検知，分類手法が提案されている．とくに教師ありの機械学習技術を用いた場合，データを逐一学習させる手法をとることで，人力による詳細な解析を要することなく，低コストで学習結果を反映させることができるという点が挙げられる [15], [1]．しかし，現在の研究では，その精度を向上させることに集中していて，計算結果に影響を与えたデータの具体的な特徴量が分からないという問題がある．これは，マルウェアであると判定した上で，種類に応じて詳細な対策を練ろうとするマルウェア解析者側からすれば，不都合である．

本研究では，その欠点を補完した機械学習モデルを提案する．その時にマルウェアの種類として，ランサムウェアのファミリ分類を行い，特定の種類にカテゴライズされるマルウェアに対して有用であることを示す．

1.2 本文書の構成

本節では本論文の構成について述べる．2 章では，ランサムウェアの現状とその対策技術について詳述する．3 章では提案手法，またそれにまつわる要素技術を記述する．4 章では評価について説明し，5 章では関連研究を概括する．6 章では考察とその後の課題を提示する．

[1] これをオンライン学習という

第2章　ランサムウェアとその対策技術の現状

本節では，ランサムウェアの現状，対策技術を述べて，本研究で注目する問題を析出する．

2.1　ランサムウェア

ランサムウェアとは，コンピュータリソースやファイルのアクセスを制限して，その解除に身代金を要求するマルウェアの一種である．具体的なアクセスの制限の方法としては，メモリのリソースを制限したり，ファイルを暗号化を行い，その復号にビットコインを要求する．ランサムウェアが端末にダウンロードされると，一般的には以下の図のような動作をする（図 2.1）[21].

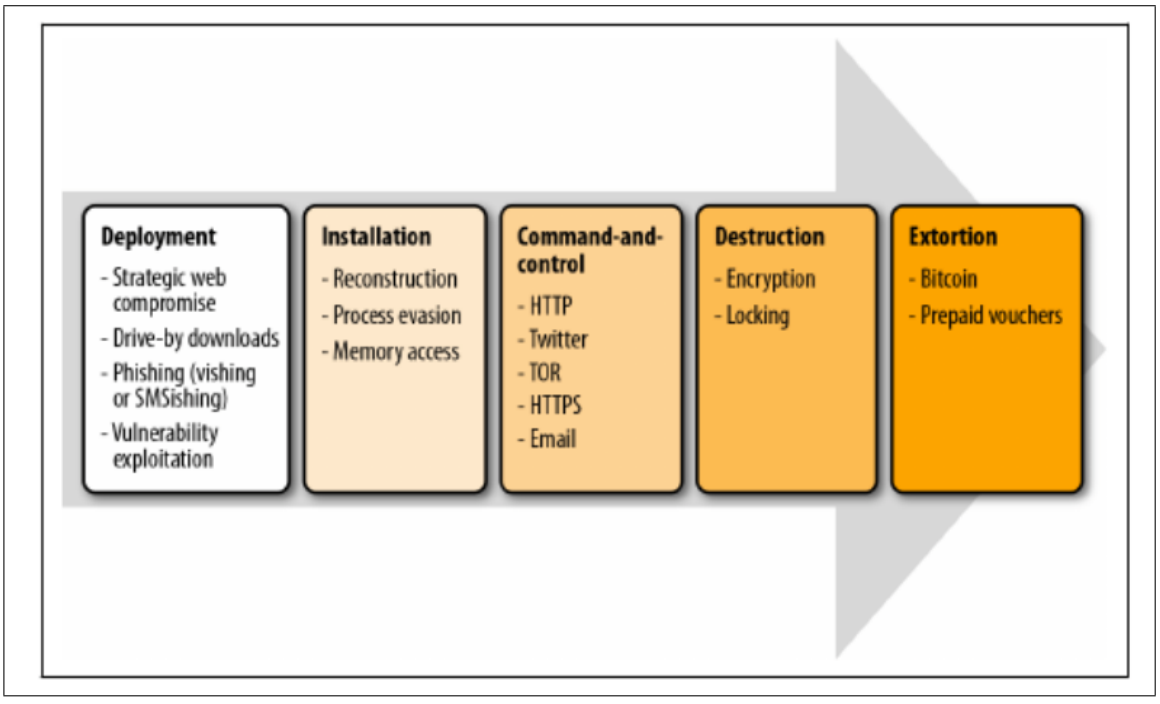

図 2.1: ランサムウェアの動作

以下に有名なランサムウェアの種類を，特徴とともに示す 2.1.

ダウンローダーが実行されてから，二次検体を C2 サーバーからダウンロードしてくるか，またはダウンローダ自身が二次検体を生成する．その後，C2 サーバーと交信をとったのち，ファイルの拡張子や，ディレクトリ情報を参考にファイルを探索し，暗号化を施す．その解

表 2.1: ランサムウェアの種類

ファミリ	種類
CryptoLocker	C2 サーバーの公開鍵を用いて暗号化
Wannacry	二次検体をダウンロードする前に C2 サーバーに接続
Petya	Eternal Blue という Windows 特有の脆弱性を使用
Badrabbit	Petya の後継．メモリへのアクセスを制限
UIWIX	Wannacry に解析妨害機能を賦与
Reveton	FBI を騙って金銭を要求
Kovter	ファイルの形式を用いず，メモリに常駐する

除には一定期間内の金銭の支払いを要求するが，払っても復号しない，または，ファイルそのものを削除するランサムウェアも存在する．例えば Ranscam は，ファイルを全部削除した上で身代金を要求する [16]．ファミリの中にも，金銭を払っても復号しないものも存在しており，身代金を払ったとはいえ，復元されるとは限らない．

また，ランサムウェアに限らず，マルウェアは，年を経てるごとに，既存のアンチウィルス製品や研究の検知を回避するような機能を盛り込むことが多い．以下の図のように??，例えば，世界的に有名になったランサムウェアの Wannacry[8] があるが [29]，これに続きこの Wannacry に解析環境検知の機能を追加した UIWIX[30]2.2，2016 年に猛威を振るった Petya[22] を改造した Badrabbit[7] などの，既存の種類をベースにした新しいランサムウェアが出現して，企業やインフラにダメージを与えている．つまり，攻撃者と解析者の，いわばいたちごっこの様相を呈している．また，マルウェアの数は年を経てるごとに指数関数的に増加していて，2017 年では 7 億個ものマルウェアが出現している．

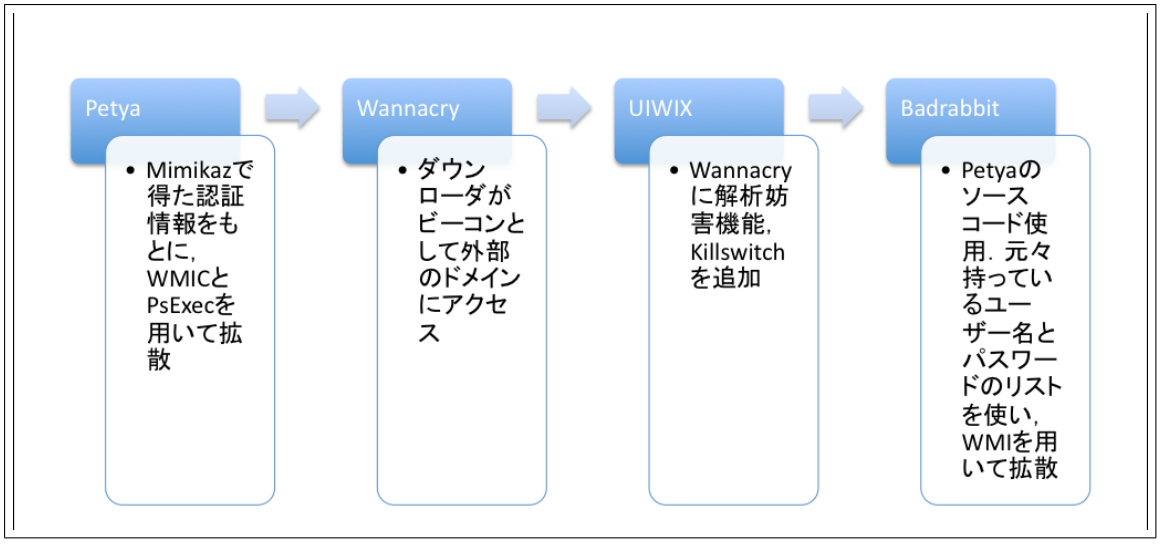

図 2.2: ランサムウェアの時系列的変遷

そのような状況下，ランサムウェアによる攻撃に対し対策を行うためには，時が経つごとにその特徴が変わりゆく検体自体を逐次詳細に解析する必要がある．そこで，既存研究や製品などでは，例えばファイルアクセスを監視する手法のように，マルウェアのファミリの特徴を見出し，それを利用して解析を自動化しようと試みている．しかし，上で述べた通り，解析手法を迂回するような方法を攻撃者側が実装して解析を遅らせようとしており，これらの手法は永続的に利用可能なものではない．また，統計によると [3]，マルウェアの数は年を経てるごとに指数関数的に増加している．人力によってこれらを網羅的に解析し，その都度攻撃の対策をするのは困難を極めている．

2.2 既存研究の問題点

以下では，前節で記したマルウェアの現況に対する，既存研究の問題点を述べ，本研究での問題意識を提示する．

2.2.1 ランサムウェア検知

マルウェア検知の研究では，その特徴を用いた検知手法が提案されている．

ランサムウェアの研究の場合，ファイルアクセスに特徴を見出した手法がある．Khazar らは，既存の動的解析環境がランラムウェアの検知，分類に特化していないという問題意識を持って，ファイルアクセスと画面の変化を監視する動的解析環境を開発した [17]．また，Andrea らは，暗号化及び削除を受けたファイルを修復する手法を提案した [5]．さらに，Khazar らはファイルアクセス監視機能を OS に賦与し，怪しい挙動をしたものに対してはプロセスごと停止させる手法を提案した [18]．

しかし，これらは単一の特徴量のみを考慮に入れていて，それを迂回しうる機能を盛り込まれたら既存の手法が通用しなくなる可能性が有る．実際に，先述の Khazar のファイルアクセスと画面の検知を監視するシステムは，ユーザーランドのバイナリの動作を主眼に置いていて，ルートキットなどによる，権限昇格などを利用してカーネルレベルの操作を掌握された時には先述の実装は無効になる．

2.2.2 機械学習を用いたマルウェア検知技術

以上の状況を考慮して，研究，産業界の製品では，機械学習技術，特に教師あり学習を用いて，ランサムウェアを含めた，未知のマルウェアに対処しようとしている．例えば産業界の場合，Cylance は，教師あり機械学習を用いてアルゴリズムを生成して，そのアルゴリズムを元にマルウェアを検知している [6]．

研究でも多様な手法が提案されている．Saxe らの研究で，エントロピーや PE ヘッダの情報を，ノードに隠れ層を追加した，深層学習の一種である，Deep Neural Netwoek を用いて分類する手法がある [26]．また，鳶山らはリカレントニューラルネットワークを用いて画

像に変換，またそれを畳み込みニューラルネットワークを用いて識別した [28]．これは偽陽性が少ないなどの報告がある．

しかし，これらの教師あり機械学習を用いたシステムは，往々にして計算結果だけを提示するのみである，すなわち，どの特徴量を見て判定したのか，具体的な判断基準を示すことができないという問題がある．これによって，当該モデルをどの程度信頼すればいいのか不明になる．マルウェアの持つ特徴は時代によって変化しており，従って教師データ自体が現実的に起こっている攻撃に追従していなければ，分類能力を下げる可能性がある．加えて，以上のようなシステムで誤検知の処理が問題になる，つまり，判断基準が不明な誤検知は実際の解析現場では不便である．そのような状況で，計算結果とその確からしさだけでは不十分で，以上に加えて実際の判断基準を詳らかにしなければならない．これに配慮して，先行研究では，Deep Neural Network に関して，計算が複雑であることが問題視されていて，計算過程を明らかにすることで意味解釈性を賦与する研究が盛んである．Zhou は畳み込みニューラルネットワークが注視している画像を可視化した [31]．また，2017 年には，畳み込みニューラルネットの隠れ層をラベル付けすることで意味解釈性を追加した [13]．これらの，Deep Neural Network の意味解釈性を実際のマルウェア検知モデルに追加した研究はまだ存在しない．加えて，実際のマルウェア検知モデルとして，線形 SVC を用いるモデルも提案されてきているが [1]，線形 SVC だからできるものであり，汎用的にすべてのモデルで信頼性を賦与することは難しい．

以上に加えて，近年では機械学習に対する攻撃手法が存在する観点からも，モデルの持つ確度の高い判断基準を詳細に提示する必要がある．例えば，Deep Neural Network の場合では，モデルと活性化関数を用いて人間には感知不能な微小なノイズを計算して，それを元のデータに付加することで検知率を低下させる性質が知られている [11]．これに関しては，動画などの，実世界での応用も多数検討されている [20] [14] [2]．もちろんマルウェアの分野でも例外ではない．2017 年現在の研究では，Deep Neural Network を使用してマルウェア検知モデルを作って，ノイズ入りのテストデータを検知させた研究 [12] や，強化学習や，GAN などの機械学習モデルを使って，アンチウイルス製品の検知を回避するようにマルウェアの部分を変更する研究も存在する [13] [9]．

本研究では，そのような教師あり機械学習のシステムの欠点を補完するべく，モデル自体の判断基準を抽出する特定のファミリの分類システムを開発することを目標とする．

第3章　解析手法

本節では，前章で提示した課題を受けて，提案手法を詳述する．

3.1　概要

前節で説明した，教師あり学習の機械学習を使ったマルウェア分類モデル自体の計算過程が複雑で，具体的に計算結果に貢献した特徴量がわからない，という課題を受けて，近似化アルゴリズムである LIME を用いて，機械学習モデルの近似を行い，判断基準を抽出する．ここでは分類器として StackNet を機械学習モデルとして用いた．

3.1.1　提案手法概観

提案手法は以下の図のように示すことができる（図 3.1）．

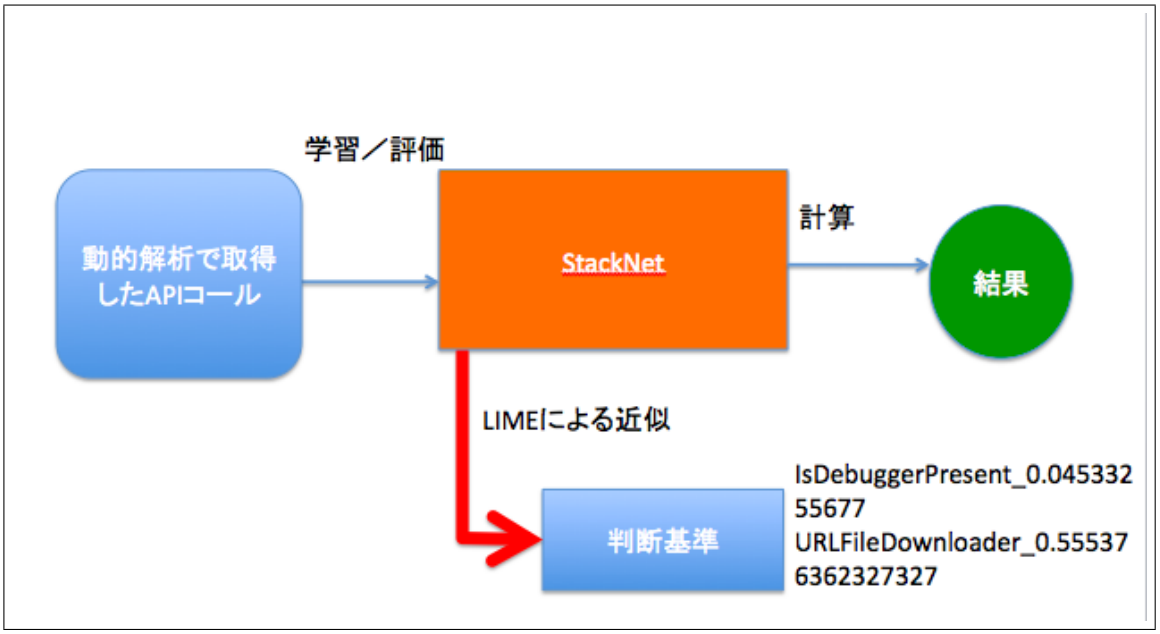

図 3.1: 提案手法の概観

計算結果を提示しながら，それと同時により結果に貢献した特徴量を抽出して提示する．

3.1.2 LIME

LIME[25] は，機械学習が計算する決定境界を近似するアルゴリズムである．LIME は，入力データをランダムに変異させて，そのデータによる判断結果を総覧する．データによっては，結果がそれぞれ違うことがある．その違う判断結果を考慮してより簡単な境界を確定する．以下の図を参照すると，青いエリアのサンプリングのエリアと黄色いサンプリングのエリアは違うことが分かる．この結果，境界の概形を把握することができる（図 3.2）．

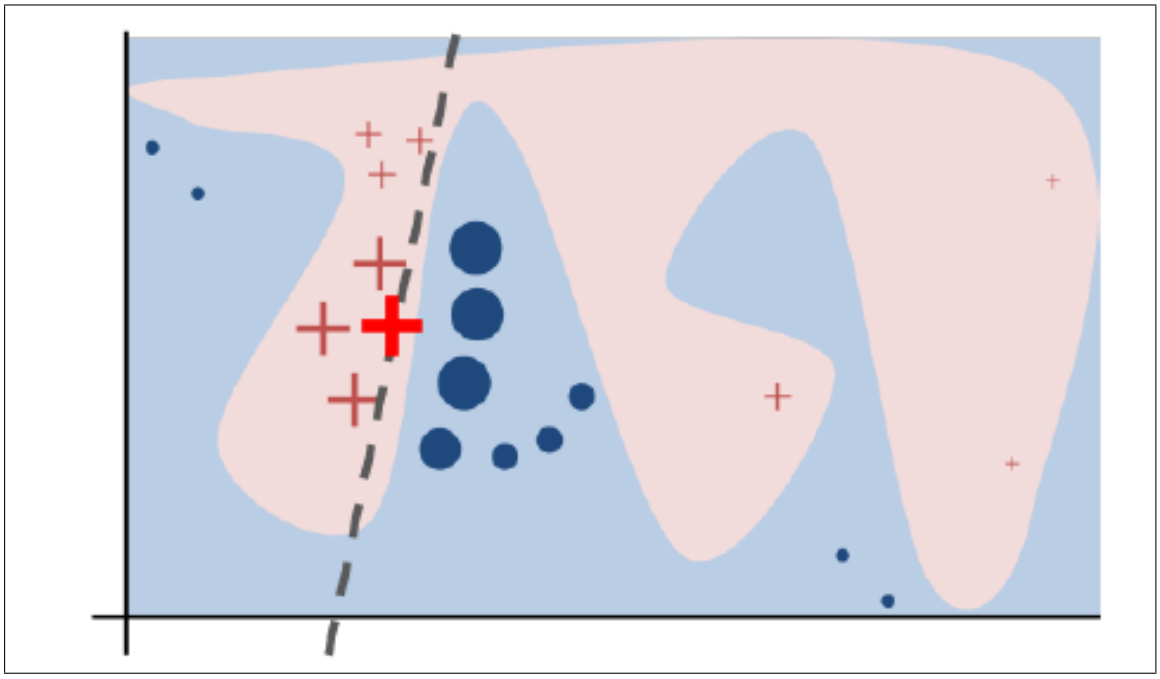

図 3.2: LIME の境界確定法

数式では，以下のように表せる．

$$\xi(x) = argmin_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{3.1}$$

x が入力，G があらかじめ持つ解釈できる線形モデルの集合，g がその一部，L が，ランダムにサンプリングした x を用いたとき，説明したい機械学習モデルの境界と，G から選んだモデルがどの程度離れているかを示す関数，Ω がモデルの複雑性である．これの和が最小になるときの g の集合を求めるのが，この式の要諦である．上式をもって選定した g をもって境界を簡便にする．

本アルゴリズムは，一つのデータごとにそのデータの中の具体的な特徴量を抽出する．本研究では，テストデータのうち，モデルが正解したテストデータだけを用いてアルゴリズムを適用させ，集計する．

3.1.3 実験環境

以下の実験環境で行う 3.1.

表 3.1: 実験環境

OS	Ubuntu 16.04.3 LTS
判断基準抽出	LIME 0.1.1.25
データセット	RISS のランサムウェア 371 検体の動的解析ログ
分類器	StackNet(scikit-learn 0.19.1, xgboost 0.7，)

3.1.4 データセット

本研究では，ケーススタディとしてランサムウェアのファミリ分類を行う．

データセットを構築するにあたり，マルウェアの挙動に着目する．そこで，Imperial College London の RISS が公開しているデータセット [27] の API コールを用いる 3.2．また，データセットを構築する上で，レジストリ操作に関する API は取り除いている．いずれの種類のマルウェアにせよ，レジストリを操作することは共通して見られる特徴であり，用いるべき特徴量として不適当であると考えられる．

評価に使用する検体は，以下である[1]．

表 3.2: ランサムウェアの種類と検体数

種類	検体数
CryptLocker	107
Kovter	64
Locker	97
Reveton	90
収集した良性ソフトウェア	79

3.1.5 StackNet

StackNet とは，分類器を組み合わせるアンサンブル学習の一種である [24]．特徴量をランダムサンプリングしたのち，分類器に渡すことで，過学習を抑制する．この手法はよく

[1]良性ソフトウェアは，http://software.informer.com/software/でトップに出ていたものを使用した

Kaggle などのコンペティションに用いられており，この手法を用いているグループは多い．組み合わせる分類器は多種多様に渡り，それぞれどの分類器を重要視するか，重みをつけることができる．ここでは，以下の分類器を用いる 3.3.

表 3.3: Stacking に使用する分類器の種類

分類器	重さ	説明
SVC	2	最大化マージンを用いて線形に決定境界を近似
RandomForest	4	決定木を多数生成するアンサンブル学習
Naive Bays	4	ベイズの定理を用いた分類器
XGBoost	7	勾配ブースティングと RandomForest を組み合わせたアンサンブル学習
Logistic Regression	2	ベルヌーイ分布を用いた統計的分類器

当該分類器は，データセットの内 9 割で学習し，他の 1 割で評価する．F 値は 0.7849548112706007 であり，テストデータ 34 個で評価した結果 28 個が正解であった．

(1) SVC

この分類器は，それぞれの訓練データから遠くなるように設定して線形に決定境界を引く分類器である．それによってモデル自体の汎化性能を向上させる．

一般的に以下の目的関数の最小値の時の w の値を，二次計画法を用いて解く．

$$f = \frac{w^2}{(w^T(x_p - x_n))^2} \tag{3.2}$$

線形で分離が難しい時には，訓練データから写像を生成してこれを線形分離しやすいように生成するカーネル SVC がある．本研究では線形 SVC を用いる．

(2) Random Forest

弱い決定木を多数生成することで行うアンサンブル学習である．決定木学習とは，訓練データの集合の要素についてばらつきを減少させることを企図して特徴量を分割する．一般に決定木学習は，質問の量が多い，つまり木が深くなればなるごど過学習に陥りやすいのであるが，Random Forest では，木を深くしても過学習が起きにくい．

$$g = 1 - \sum_{i=1}^{K} p^2(C_i|t) \tag{3.3}$$

また，ジニ係数をみることで特徴量の重要性をみることができる．特徴量の取捨選択のときにこれを用いることもある．

(3) Naive Bays

ベイズの定理をもとにしているもの.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{3.4}$$

この時の $P(A)$ は定数としてみなせる．$P(A|B)$ を，以下のシンプルな式に限定して解く.

$$P(A|B) = \prod_{i=1}^{N} P(x_i|Y) \tag{3.5}$$

(4) XGBoost

勾配ブースト決定木と Random Forest のアンサンブル学習である．勾配ブースト決定木は，逐次木の集団での観測値と予測値を考慮して，弱い決定木を逐次生成する.

(5) Logistic Regression

ベルヌーイ分布を用いた確率統計的分類器である．予測変数をロジット変換することで確率分布にして，説明変数の式を探索する．活性化関数には，シグモイド関数を用いる.

$$\phi(z) = \frac{1}{1 + e^{-}z} \tag{3.6}$$

回帰という名を冠しているが，教師あり学習でも用いられる.

第4章　解析結果

本節では，前章で提示した手法を用いて実験を実施し，提案手法についての評価を行う．

4.1　評価方法

ここでは二種類の評価方法を導入する 4.1.

表 4.1: 評価手法

評価手法名	説明
評価手法 (1)	抽出された API を機能別に分類
評価手法 (2)	抽出された API を用いてルールベース検知システムを作成し評価

4.1.1　評価手法 (1)

評価手法 (1) では，ランサムウェア独特の特徴を考慮した項目を設定する．ランサムウェアには，その機能上，三つの不動点たる特徴が存在する．つまり，暗号化を行う機能，ファイルにアクセスする機能，インターネット接続を行う機能である。これらは，実際に動的解析の結果上では，Windows API によってその痕跡を確認できることが多い[1]．そこで，暗号化に関する API，インターネット接続に関する API，ファイル操作に関する API，その他に分けて，出現した特徴量を評価する．

4.1.2　評価手法 (2)

評価手法 (2) では，実際にルールベースのシステムを構築する．そのために特徴量と重みを紐付けた辞書を定義する．以下の例の様に，LIME では，抽出された特徴量の他に，どの程度影響を与えたのか示す数値も提示する．この機能を利用し，ルールベースのための辞書を構築する．以下，このように作成した辞書をルールベース辞書と呼称する．

以下の図の様にルールベース辞書を定義する 4.1.

[1]暗号化を行う関数に関しては，Win32/CTBLocker や Win32/TorrentLocker のように，現存のライブラリを使用することなく実装するものも存在する

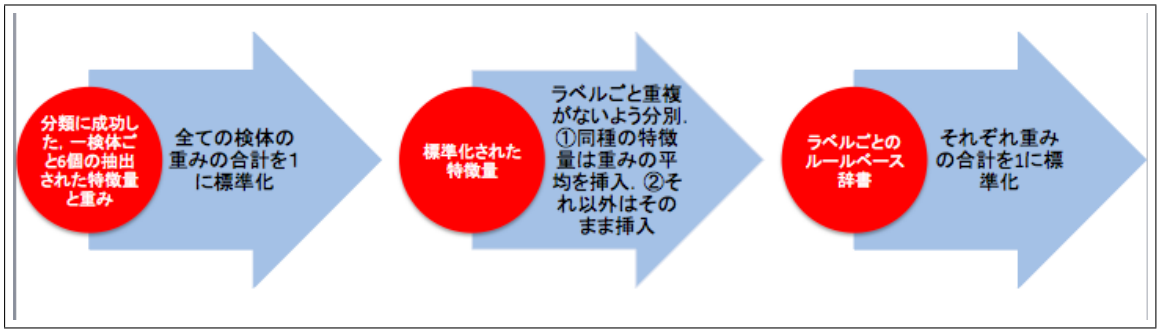

図 4.1: 提案手法の概観

最初に，当該分類器で分類に成功した検体を抽出する．一検体ごとに抽出した特徴量 6 個を，合計が 1 になるように標準化する．次に，標準化された検体ごとの判断基準を，ラベルごとに分けて，重複がないように整理する．もし，同じ特徴量が複数出現した際には，それぞれの重みを平均を取って挿入する．それ重複のない特徴量はそのまま挿入する．この時できたラベルごとの辞書の重みの合計を 1 に標準化する．以下が辞書の一例である 4.2.

表 4.2: ルールベース辞書

特徴量	重み
CertOpenStore	0.06565736054979383
CryptAcquireContextA	0.09746892921246786
RtlAddVectoredExceptionHandler	0.059140259652749355
SetWindowsHookExA	0.062277129531377914
NtSetContextThread	0.04105626155591237
FindResourceExW	0.09923948440933845
CoInitializeSecurity	0.06568748878515018
NtTerminateThread	0.08529459411799258
NtAllocateVirtualMemory	0.08729026670236598
SetFilePointerEx	0.04822739511977077
VirtualFreeEx	0.04060162520223443
CertOpenSystemStoreA	0.03968237708736168
NtCreateMutant	0.16457576743484909
RtlDecompressBuffer	0.043801060638635585

こうしてできたラベルごとのルールベース辞書を用いて同じテストデータを評価する．それぞれのラベルごとに，テストデータの特徴量の中から，ルールベース辞書にある特徴量が何個存在するかカウントする．そして，それぞれの個数と，特徴量の重みをかけ，辞書の特徴量ごとに合計を出す．これをすべてのラベルで行い，より合計が高いものをルールベース

での判断結果と定義する.

4.2 評価結果

以下に評価を掲示する.

4.2.1 評価手法 (1) 抽出された API を機能別に分類

以下がラベルごとの評価結果である 4.3, 4.4, 4.5, 4.6.

表 4.3: CryptLocker

分類	暗号化 API	インターネット API	ファイル操作 API	その他
個数	1	1	1	11
API1	CryptAcquireContextA	RtlDecompressBuffer	SetFilePointerEx	省略
API2	なし	なし	なし	省略

表 4.4: Kovter

分類	暗号化 API	インターネット API	ファイル操作 API	その他
個数	1	2	2	8
API1	CryptAcquireContextA	InternetCloseHandle	SetFilePointerEx	省略
API2	なし	InternetOpenUrlW	SetFileAttributesW	省略

表 4.5: Locker

分類	暗号化 API	インターネット API	ファイル操作 API	その他
個数	1	1	2	16
API1	CryptAcquireContextA	InternetOpenUrlW	GetFileSize	省略
API2	なし	なし	SetFileAttributesW	省略

以上の区分けをすると，ランサムウェアとしての特徴量を抽出していることが定性的にわかる.

表 4.6: Reveton

分類	暗号化 API	インターネット API	ファイル操作 API	その他
個数	2	2	2	10
API1	CryptAcquireContextA	DnsQuery_W	SetFilePointerEx	省略
API2	CryptGenKey	InternetCrackUrlW	SetFileAttributesW	省略

4.2.2 評価手法 (2) 抽出された API を用いてルールベース検知システムを作成し評価

前節で提示した評価方法をもって，同じテストテータを評価した 4.7.

表 4.7: 評価手法 (2)

種類	正解数
StackNet	28
LIME	23

5 個ほど検知することができなかったが，おおよそ 8 割弱のテストデータを上記手法で検知することができた.

第5章　結論

本節では結論と，今後の課題を示す.

5.1　本研究の貢献

本研究は，機械学習モデルが，具体的にどの特徴量を重きを置いてみたのか不明である，という問題意識から，機械学習に近似アルゴリズムを適用したモデルを開発し、実際にランサムウェア分類ができるかどうか実験した．最初に，API からランサムウェア独特の関数である，暗号化，インターネット接続，ファイル操作を抽出することができたか確認する評価を実施し，以上三つの関数を抽出することができた．それに加えて，当該特徴量が実際に有用なのかを，抽出した特徴量をもとにルールベースの分類器を実装し，同じテストデータを用いて分類を行った．その結果，8 割の精度で，LIME を使用して分類能力を復元することができた.

5.2　今後の課題

今後，ランサムウェアそれぞれへのケーススタディをして、より有用な特徴量を提示して現場のトリアージに資することを目指していきたい.

謝辞

本研究を進めるにあたり, ご指導いただきました慶應義塾大学環境情報学部教授 村井純博士, 同学部教授 中村修博士, 武田圭史博士, Rodney D. Van Meter III 博士, 楠本博之博士に感謝申し上げます．特に武田博士には，学部が違うのに研究会で所属を了承いただきました．先生がいなければ私はここで卒論を書くことはありませんでした．また推薦書を書いていただき，感謝を申し上げます．

半期の間卒論をご指導いただきました，国立研究開発法人情報通信研究機構教授の小林和真博士に感謝申し上げます．卒論のみならず，外部からの案件にも御誘い頂き，知見が広がりました．重ねて感謝申し上げます．

元も含めた，Sigma，ISC の皆様方に感謝申し上げます．特に，澤井優作氏，鈴木恒平氏，中島春香女史，小林怜央氏，柴田彩香女史，丸林栞女史，幅野莞佑氏，宇野弘明氏，加藤太陽氏，藤田玲央氏，梶原夢華女史，Korry Luke 氏，Ko You Liang 氏，Aaron Tang 氏，Andrey Ferrryan 氏，Flo Costa 氏に感謝を申し上げます．学部よりも居心地が良くて，もう少しいたいななどと思いましたが，何も生産しない老害がいてもしょうがないし，次にやりたいこともあるので，今年で失礼させていただきます．

卒論係の皆さまに感謝します．学部の関係上，いろいろ融通きかせてくれて頭が上がりません．

黒米祐馬氏に感謝します．キャンプでのサンドボックス開発のみならず，研究と発表の作法，低レイヤのこと，インターネット界隈の深い闇などをたくさん教えていただけました．重ねて感謝申し上げます．

永山翔太博士に感謝申し上げます．SNS で卒論について相談に乗ってくださいました．重ねて感謝申し上げます．

マルウェア解析などのほかにも様々な面白いセキュリティ技術を教えてくださいました，2016 年度セキュリティキャンプの講師の方々，生徒の方々に感謝申し上げます．授業を受けて，もっとセキュリティを触っていきたいと思いました．特に西田耀女史，仲松祟斗氏，長谷川千広氏に感謝申し上げます．あのときのグループワーク，いまだに思い出されます．また，太中公幸氏，坂田成史氏，河野晋策氏にも感謝申し上げます．最終日での居酒屋での下衆すぎる会話，いまだに思い出されます．異性への ROP を使った exploitation もほどほどに．

インターンでお世話になった株式会社メディヴァの方々に感謝申し上げます．研究会と並行しての勤務は過酷を極めましたが，その忙しさもいい経験です．

英語を上達させる機会をくださった，法学部の英語インテの教授，生徒の方々に感謝申

し上げます．特に，藤澤翔氏，田中大貴氏，武石将大氏，萱野史菜女史，清水潔之氏，安藤洋陽女史，雨宮千華女史に感謝申し上げます．みんなでライブに行って，その後新大久保のホットク屋で将来のことについて語ったことが未だにいい思い出です．また，近藤康裕博士にも感謝申し上げます．英語試験に向け，拙い論述答案を何度も添削していただきました．

最後に，この性格が曲がったぼんくらを励まし，応援してくれた，家族の皆様に感謝申し上げます．

This one’s dedicated to all the hackers.

参考文献

[1] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of android malware in your pocket, 02 2014.

[2] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, Vol. abs/1707.07397, , 2017.

[3] AV-TEST. AV-TEST The Independent IT-security Institute(Cited 12th December 2017). https://www.av-test.org/en/statistics/malware/.

[4] Barun. Reversing the petya ransomware with constraint solvers(Cited 12th December 2017). https://0xec.blogspot.jp/2016/04/reversing-petya-ransomware-with.html.

[5] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: A self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Computer Security Applications Conference*. ACM, 2016.

[6] Cylance. Cylance PROTECT(Cited 12th December 2017). https://blog.cylance.com/cylanceprotect-is-the-first-signature-less-next-generation-antivirus-to-be-certified-by-av-test.

[7] Endgame. BadRabbit Technical Analysis(Cited 12th December 2017). https://www.endgame.com/blog/technical-blog/badrabbit-technical-analysis.

[8] Endgame. WCry/WanaCry Ransomware Technical Analysis(Cited 12th December 2017). https://www.endgame.com/blog/technical-blog/wcrywanacry-ransomware-technical-analysis.

[9] Hyrum Anderson Anant Kharkar Bobby Filar and Phil Roth. Evading machine learning malware detection. *BlackHat USA*, 2017.

[10] Joseph Gardiner and Shishir Nagaraja. On the security of machine learning in malware c&c detection: A survey. *ACM Comput. Surv.*, Vol. 49, No. 3, pp. 59:1–59:39, 2016.

[11] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, December 2014.

[12] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick D. McDaniel. Adversarial perturbations against deep neural networks for malware classification. *CoRR*, Vol. abs/1606.04435, , 2016.

[13] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on GAN. *CoRR*, Vol. abs/1702.05983, , 2017.

[14] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Query-Efficient Black-box Adversarial Examples. *ArXiv e-prints*, December 2017.

[15] Heju Jiang, Jasvir Nagra, and Parvez Ahammad. Sok: Applying machine learning in security - A survey. *CoRR*, Vol. abs/1611.03186, , 2016.

[16] Kaspersky. 身代金を払っても払わなくても酷い目に遭わせるランサムウェア「Ranscam」(Cited 12th December 2017). https://blog.kaspersky.co.jp/ranscam-ransomware/11991/.

[17] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. Unveil: A large-scale, automated approach to detecting ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, pp. 757–772, Austin, TX, 2016. USENIX Association.

[18] Amin Kharraz and Engin Kirda. Redemption: Real-time protection against ransomware at end-hosts. In *Proceedings of the 20th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 9 2017.

[19] Youngjoon Ki, Eunjin Kim, and Huy Kang Kim. A novel approach to detect malware based on api call sequence analysis. *Int. J. Distrib. Sen. Netw.*, Vol. 2015, pp. 4:4–4:4, January 2015.

[20] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, Vol. abs/1607.02533, , 2016.

[21] Allan Liska and Timothy Gallo. Ransomware : Defending against digital extortion. *O'Reilly Media, Incorporated*, p. 6, 2016.

[22] Malwarebytes. Petya Taking Ransomware To The Low Level(Cited 12th December 2017). https://blog.malwarebytes.com/threat-analysis/2016/04/petya-ransomware/.

[23] Micheal Shikorski and Andrew Honig. Practical Malware Analysis: A Hands-On Guide to Dissecting Malicious Software. *No Starch Press*, p. 1, 2012.

[24] Sebastian Raschka. Mlxtend, April 2016.

[25] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, Vol. abs/1602.04938, , 2016.

[26] Joshua Saxe and Konstantin Berlin. Deep neural network based malware detection using two dimensional binary program features. In *Proceedings of the 2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, MALWARE '15, pp. 11–20, Washington, DC, USA, 2015. IEEE Computer Society.

[27] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C. Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *CoRR*, Vol. abs/1609.03020, , 2016.

[28] Shun Tobiyama, Yukiko Yamaguchi, Hajime Shimada, Tomonori Ikuse, and Takeshi Yagi. Malware detection with deep neural network using process behavior. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Vol. 2, pp. 577–582, 2016.

[29] TrendMicro. 「WannaCry」の残した被害と教訓、2017 年上半期の脅威動向を分析 (Cited 12th December 2017). http://blog.trendmicro.co.jp/archives/15925.

[30] TrendMicro. ランサムウェア「UIWIX」など「WannaCry ／ Wcry」の模倣犯が連続して出現 (Cited 12th December 2017). http://blog.trendmicro.co.jp/archives/14934.

[31] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *CoRR*, Vol. abs/1711.05611, , 2017.

[32] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, Vol. abs/1512.04150, , 2015.

付 録A　実際に抽出した判断基準

```
label 0
NtDuplicateObject\_0.0632036463575
getaddrinfo\_0.0671723299405
InternetSetStatusCallback\_0.162051241779
NtReadVirtualMemory\_0.126457515505
LdrUnloadDll\_0.0471642540513
VirtualFreeEx\_0.0586373775965
GetDiskFreeSpaceExW\_0.138883585072
LookupPrivilegeValueW\_0.0444914068792
InternetOpenUrlW\_0.103889737199
SetFilePointerEx\_0.0624778074895
LdrLoadDll\_0.0449317047301
SetFileAttributesW\_0.0806393934025
label 2
CertOpenStore\_0.06565736054979383
CryptAcquireContextA\_0.09746892921246786
RtlAddVectoredExceptionHandler\_0.059140259652749355
SetWindowsHookExA\_0.062277129531377914
NtSetContextThread\_0.04105626155591237
FindResourceExW\_0.09923948440933845
CoInitializeSecurity\_0.06568748878515018
NtTerminateThread\_0.08529459411799258
NtAllocateVirtualMemory\_0.08729026670236598
SetFilePointerEx\_0.04822739511977077
VirtualFreeEx\_0.04060162520223443
CertOpenSystemStoreA\_0.03968237708736168
NtCreateMutant\_0.16457576743484909
```

```
RtlDecompressBuffer\_0.043801060638635585
label 5
CreateToolhelp32Snapshot\_0.09103020080979546
GetVolumePathNamesForVolumeNameW\_0.060164927283747636
NtGetContextThread\_0.012486619597247726
CryptAcquireContextA\_0.15725763084863198
GetShortPathNameW\_0.10361133988202284
VirtualFreeEx\_0.04026329337460649
SetFilePointerEx\_0.11929935668600537
InternetCloseHandle\_0.1015866099619152
NtDelayExecution\_0.07273127914595034
ShellExecuteExW\_0.0805407784292588
InternetOpenUrlW\_0.06985645487716703
SetFileAttributesW\_0.0649912439890462
NtFreeVirtualMemory\_0.026180265114604934
label 6
Process32NextW\_0.10460913825605055
CreateToolhelp32Snapshot\_0.05848916625359279
NtReadVirtualMemory\_0.
CryptAcquireContextA\_0.04657534018855659
GetFileSize\_0.05999235318700823
LoadResource\_0.041953443778324245
FindResourceExW\_0.046256478386491794
CoInitializeSecurity\_0.04106900071070243
GetVolumePathNamesForVolumeNameW\_0.025528759153416996
NtTerminateThread\_0.04015248430111379
InternetOpenUrlW\_0.04116016744004671
NtAllocateVirtualMemory\_0.04949477530080154
GetComputerNameA\_0.03651255124985349
CreateDirectoryW\_0.05253796486989479
NtQueryAttributesFile\_0.04414445834123835
GetKeyState\_0.029576721780765793
MessageBoxTimeoutA\_0.0732514470702257
```

```
SetFileAttributesW\_0.039122615792850465
NtOpenProcess\_0.059753293296265275
NtGetContextThread\_0.0455846803614533
label 9
GetVolumePathNamesForVolumeNameW\_0.03959556691087611
CreateToolhelp32Snapshot\_0.05089831593716131
SetFilePointerEx\_0.07005514999775289
RtlAddVectoredExceptionHandler\_0.004489426260969539
CryptAcquireContextA\_0.061300677293984966
LdrUnloadDll\_0.11931091468829742
CreateServiceA\_0.02132018793847518
OpenSCManagerW\_0.026908562489358398
NtSetInformationFile\_0.04404276438906645
InternetCrackUrlW\_0.055363904321261224
LdrGetProcedureAddress\_0.08153885061665621
FindResourceW\_0.12618614400357414
DnsQuery\_W\_0.09487863009185622
ShellExecuteExW\_0.03613691971504754
SetFileAttributesW\_0.08989347887622046
NtOpenProcess\_0.078080506469442
```

付 録B　判断基準抽出に使用した検体のラベルの内訳

表 B.1: ランサムウェアの種類と検体数

種類	検体数
CryptLocker	7
Kovter	4
Locker	5
Reveton	10
収集した良性ソフトウェア	2

付 録C　実際に抽出した検体ごとの判断基準

```
NtReadVirtualMemorydash0.178980943402,
GetFileSizedash0.212407819571,
FindResourceExWdash0.150211143775,
NtTerminateThreaddash0.140211333626,
CreateDirectoryWdash0.174836091847,
NtQueryAttributesFiledash0.143352667779

CryptAcquireContextAdash0.205141443998,
NtTerminateThreaddash0.133225183575,
SetFilePointerExdash0.0999729971617,
CertOpenSystemStoreAdash0.0945126864155,
NtCreateMutantdash0.373029728196,
RtlDecompressBufferdash0.0941179606541

CryptAcquireContextAdash0.130061817505,
InternetCrackUrlWdash0.12697329849,
LdrGetProcedureAddressdash0.21677462105,
DnsQuery\_Wdash0.204342680778,
SetFilePointerExdash0.150921205292,
NtOpenProcessdash0.170926376885

NtDuplicateObjectdash0.126407292715,
getaddrinfodash0.134344659881,
NtReadVirtualMemorydash0.252915031009,
VirtualFreeExdash0.117274755193,
InternetOpenUrlWdash0.207779474397,
SetFileAttributesWdash0.161278786805

CryptAcquireContextAdash0.114493463413,
InternetCrackUrlWdash0.127819572617,
LdrGetProcedureAddressdash0.210998120184,
DnsQuery\_Wdash0.218102392528,
SetFilePointerExdash0.143297775466,
NtOpenProcessdash0.185288675793

Process32NextWdash0.339702866601,
NtGetContextThreaddash0.130822565488,
CryptAcquireContextAdash0.099963112092,
NtTerminateThreaddash0.120567299018,
GetComputerNameAdash0.118569166454,
MessageBoxTimeoutAdash0.190374990347

CertOpenStoredash0.13675846062,
SetWindowsHookExAdash0.130854864749,
FindResourceExWdash0.192722513722,
CoInitializeSecuritydash0.132890821955,
NtTerminateThreaddash0.208382594731,
NtAllocateVirtualMemorydash0.198390744223

GetVolumePathNamesForVolumeNameWdash0.0805213923167,
SetFilePointerExdash0.216887745669,
CryptAcquireContextAdash0.211764458444,
LdrGetProcedureAddressdash0.106242571178,
DnsQuery\_Wdash0.250610528383,
```

```
SetFileAttributesWdash0.133973304009

CryptAcquireContextAdash0.204668687893,
NtTerminateThreaddash0.132266883407,
SetFilePointerExdash0.103028435726,
CertOpenSystemStoreAdash0.0884975855947,
NtCreateMutantdash0.38128650834,
RtlDecompressBufferdash0.09025189904

GetVolumePathNamesForVolumeNameWdash0.102101873386,
SetFilePointerExdash0.224341300377,
CryptAcquireContextAdash0.202476947039,
DnsQuery\_Wdash0.25269742438,
ShellExecuteExWdash0.0833356206978,
SetFileAttributesWdash0.13504683412

CreateToolhelp32Snapshotdash0.117376986875,
CryptAcquireContextAdash0.10993846029,
FindResourceWdash0.290998809996,
LdrGetProcedureAddressdash0.139674641394,
DnsQuery\_Wdash0.199008107624,
SetFilePointerExdash0.143002993821

SetFilePointerExdash0.170557458155,
CryptAcquireContextAdash0.210428077264,
NtDelayExecutiondash0.14108102675,
ShellExecuteExWdash0.209052160051,
InternetOpenUrlWdash0.142814040419,
SetFileAttributesWdash0.126067237362

CreateToolhelp32Snapshotdash0.18993500733,
GetVolumePathNamesForVolumeNameWdash0.0829009091343,
NtReadVirtualMemorydash0.261958472953,
CryptAcquireContextAdash0.242114566182,SetFileAttributesWdash0
    .127044969066,
GetKeyStatedash0.0960460753342

CertOpenStoredash0.137509479149,
CryptAcquireContextAdash0.13786762172,
SetWindowsHookExAdash0.131537088934,
NtTerminateThreaddash0.199153012255,
FindResourceExWdash0.204992483388,
NtAllocateVirtualMemorydash0.188940314554

CryptAcquireContextAdash0.122510826122,
InternetCrackUrlWdash0.126852634845,
LdrGetProcedureAddressdash0.207196744278,
DnsQuery\_Wdash0.214387749585,
SetFilePointerExdash0.146782552054,
NtOpenProcessdash0.182269493116

CryptAcquireContextAdash0.126664628061,
SetWindowsHookExAdash0.127419350801,
NtTerminateThreaddash0.199526105514,
CoInitializeSecuritydash0.151077786939,
FindResourceExWdash0.235140490335,
NtAllocateVirtualMemorydash0.160171638349

CryptAcquireContextAdash0.351336708139,
RtlAddVectoredExceptionHandlerdash0.124468234507,
NtSetContextThreaddash0.0864081493948,
VirtualFreeExdash0.0854513090864,
CertOpenSystemStoreAdash0.0675396298387,
NtCreateMutantdash0.284795969034

CryptAcquireContextAdash0.1224301491,
```

```
InternetCrackUrlWdash0.129054948569,
LdrGetProcedureAddressdash0.205051041283,
DnsQuery\_Wdash0.214206751031,
SetFilePointerExdash0.139855594411,
NtOpenProcessdash0.189401515606

NtGetContextThreaddash0.165236590529,
CryptAcquireContextAdash0.146126834555,
LoadResourcedash0.136237668648,
CoInitializeSecuritydash0.133365569227,
InternetOpenUrlWdash0.133661619838,
MessageBoxTimeoutAdash0.285371717202

LdrUnloadDlldash0.275143792276,
RtlAddVectoredExceptionHandlerdash0.0103530994613,
CreateServiceAdash0.0491666447848,
NtSetInformationFiledash0.101567348201,
OpenSCManagerWdash0.0620540371127,
SetFileAttributesWdash0.501715078164

NtGetContextThreaddash0.0242210110162,
CryptAcquireContextAdash0.479824233203,
VirtualFreeExdash0.128015051615,
InternetCloseHandledash0.197053364189,
ShellExecuteExWdash0.120102980653,
NtFreeVirtualMemorydash0.0507833593239

SetFilePointerExdash0.149128388345,
CryptAcquireContextAdash0.135736870483,
LdrGetProcedureAddressdash0.204753624821,
DnsQuery\_Wdash0.205077926672,
SetFileAttributesWdash0.130506274011,
NtOpenProcessdash0.174796915667

NtReadVirtualMemorydash0.184843429616,
CryptAcquireContextAdash0.116781890414,
GetFileSizedash0.177224954229,
CreateDirectoryWdash0.166382611781,
NtAllocateVirtualMemorydash0.160727039069,N
tOpenProcessdash0.194040074892

LdrUnloadDlldash0.0943285081025,
InternetSetStatusCallbackdash0.324102483557,
GetDiskFreeSpaceExWdash0.277767170143,
LookupPrivilegeValueWdash0.0889828137583,
SetFilePointerExdash0.124955614979,
LdrLoadDlldash0.0898634094601

CertOpenStoredash0.140284978903,
SetWindowsHookExAdash0.134469407738,
FindResourceExWdash0.202593223152,
CoInitializeSecuritydash0.130774535943,
NtTerminateThreaddash0.204526463844,
NtAllocateVirtualMemorydash0.18735139042

SetFilePointerExdash0.139774790553,
CryptAcquireContextAdash0.122880559073,
LdrGetProcedureAddressdash0.213607459443,DnsQuery\_Wdash0
   .210769302132,
SetFileAttributesWdash0.135278627889,N
tOpenProcessdash0.177689260911

CreateToolhelp32Snapshotdash0.176576492897,
GetVolumePathNamesForVolumeNameWdash0.116705354494,
CryptAcquireContextAdash0.248603894305,
ShellExecuteExWdash0.125080277243,
```

```
InternetOpenUrlWdash0.128195090588,
SetFilePointerExdash0.204838890473

GetShortPathNameWdash0.200980848751,
VirtualFreeExdash0.0281869796521,
ShellExecuteExWdash0.170682820779,
CryptAcquireContextAdash0.281310319723,
SetFilePointerExdash0.318839031095
```

付 録D　著者研究発表

野間口圭，黒米祐馬，武田圭史，村井純，FaceNet に対する Adversarial Examples による意図的誤検出および誤認識, 第 79 回情報処理学会. 2017.