HTTPヘッダフィールドの可変性に基づくマルウェア感染端末の特定

水野 翔[1,a)]　畑田 充弘[1,2,b)]　森 達哉[1,c)]　後藤 滋樹[1,d)]

概要：マルウェアによる脅威は依然として顕著である．感染の対策を講じていても，日々現れる新種マルウェアによって端末が感染してしまう危険性は十分にある．本研究の目的は，組織のインターネット・ゲートウェイで観測される大量の通信データを解析し，マルウェアに感染している端末を検知することである．そのようなシステムを実現するために，高精度な弁別手法を提案すると共に，悪性通信と良性通信の弁別に貢献する特徴を明らかにする．主要なアイデアは，まず初めにHTTPパケットのヘッダに記録された膨大な情報の可変性に着目し，クラスタリングを応用したテンプレート化手法によって情報を集約する．その後に機械学習を適用することで，悪性通信と良性通信の弁別と，特徴として貢献したテンプレートの抽出を行う．実データを利用した評価実験では，90.5%の高精度で誤検知率を低く維持しながら悪性通信と良性通信の弁別が可能であることを示した．また，240,858種類あった特徴量を最終的に5,502種類にまで削減することに成功した．

キーワード：マルウェア，悪性通信，機械学習，テンプレート

Detecting malware-infected hosts based on the variability of HTTP header fields

Sho Mizuno[1,a)]　Mitsuhiro Hatada[1,2,b)]　Tatsuya Mori[1,c)]　Shigeki Goto[1,d)]

Abstract: Damage caused by malware has been a significant problem that needs to be addressed. Even if we take countermeasures for infection, there would be new malware which cannot be detected by known signatures. The purpose of this study is to identify devices that are likely infected with malware by analyzing the volume of traffic observed in Internet gateway of an organization. It should cover new malware. To realize such a system, we propose a method to extract features which contribute to discriminate malicious traffic from legitimate one with high precision. Our key idea is to make use of various information in HTTP headers and aggregate the information by creating templates by applying clustering technique. We adopt a machine learning algorithm to classify observed traffic into two classes: legitimate and malicious. Through the extensive experiments using test data, we demonstrate that our methodology can achieve up to 90.5% precision and keep low false positive rate in discriminating between malicious traffic and legitimate one. We also succeeded in reducing the number of features from 240,858 to 5,502.

Keywords: malware, malicious traffic, machine learning, template

1. はじめに

近年，新種のマルウェアの増加が著しい．アンチウィルスベンダであるMcAfeeのレポートでは，2014年の初めに2億種類を超える程度の数だったマルウェアが，2015年

[1] 早稲田大学 基幹理工学研究科
Graduate School of Fundamental Science and Engineering, Waseda University
[2] NTTコミュニケーションズ株式会社
NTT Communications Corporation
[a)] mizuno@nsl.cs.waseda.ac.jp
[b)] m.hatada@nsl.cs.waseda.ac.jp
[c)] mori@nsl.cs.waseda.ac.jp
[d)] goto@goto.info.waseda.ac.jp

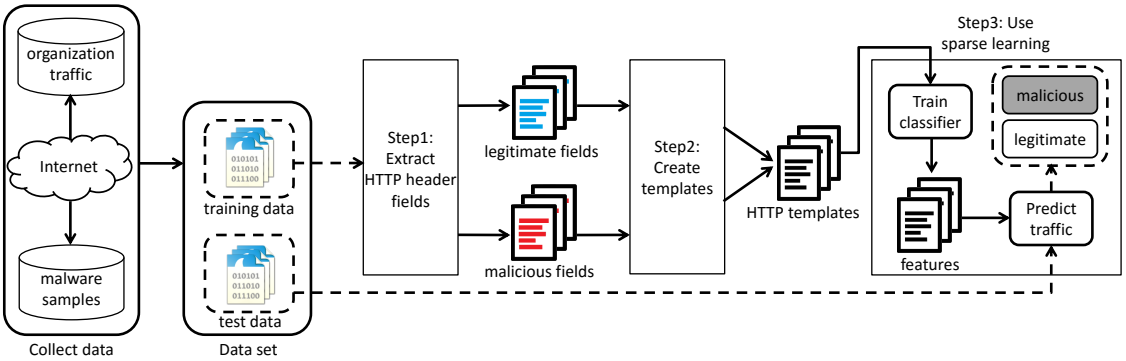

図 1　本研究のシステム概要

末にはその 2 倍以上の数である 5 億種類に迫っているという [1]．さらに，同レポートでは Mac を対象としたマルウェアの増加に対しての調査も行っており，2014 年の初めに 5,000 程度だった検体の種類が，2015 年末には 7 万強にまで増加しているという．2016 年 4 月には OS X のシェアが初めて 9%に達した [2] こともあり，近年の Mac ユーザの増加に比例してマルウェアの種類も増加を辿っていることがわかる．

ESET の発表では，2016 年は IoT（Internet of Things）を狙った攻撃の増加が予測されている [3]．実際の IoT に関するサイバー攻撃としては，世界 105 ヶ国に設置された約 2 万 5,000 台の防犯カメラがボット化し，DDoS 攻撃の踏み台になるという事案が記憶に新しい [4]．ここまで大規模な IoT を利用した攻撃はこの事案が最初だという．このように，IoT のセキュリティに関してもマルウェアの存在は無視できないものになりつつある．

マルウェアの攻撃対象は OS，端末を問わず多様化が進行しており，それらの攻撃への対応が追いつかずに端末が感染してしまうことは避けられない状況にある．そこで次に重要となるのは，いかにして感染端末を早期発見できるかということである．

本研究の目的は，インターネット・ゲートウェイで観測した大量のトラフィックを解析し，マルウェア感染端末を早期発見することである．そのようなシステムを実現するために，検知率の高さと誤検知率の低さを併せ持つ弁別手法を提案すると共に，弁別に貢献する特徴を明らかにする．主要なアイデアとしては，まず最初に HTTP パケットのヘッダに記録された膨大な数のフィールドの可変性に着目し，テンプレート化による情報の集約を行う．その後に機械学習を適用することで，悪性通信のヒューリスティックな検知を実現する．HTTP パケットに着目した理由としては，マルウェアの活動において HTTP プロトコルが使用されることが多いという点が挙げられる [5]．本論文の貢献は以下である．

- HTTP ヘッダ内の情報のみを利用することで，拡張性のあるシステムを実現し，悪性通信と良性通信を高精度かつ低誤検知率で弁別することが可能であることを示した．このシステムによって，マルウェア感染端末の早期発見とその対処が期待できる．
- HTTP テンプレートを作成することで，機械学習に用いる特徴量の削減と，それによる弁別精度の大幅な向上が示された．テンプレート化は特徴量を設定する上で非常に有効に機能することが定量的に明らかとなった．
- 識別器に入力した HTTP テンプレートの内，各通信の弁別に特に貢献したものを抽出し，分析した．これにより，IDS/IPS や WAF の新しいシグネチャとして，このテンプレートを応用することが期待される．

本論文の構成は以下の通りである． 2 章では使用するデータセットの説明と，特徴のテンプレート化及び悪性通信の弁別手法を述べる． 3 章でテンプレート化の結果と各通信の弁別結果を示す． 4 章で関連研究を紹介し，最後に 5 章で結論と今後の課題をまとめる．

2. テンプレート化と弁別手法

図 1 はシステム全体の概要である．本研究は主に 3 つのステップで構成されている．以下にデータセットと各ステップの詳細な説明を述べる．

2.1 データセット

本研究では，マルウェアによる悪性通信と一般のユーザが行う通信の弁別を行うことで，マルウェア感染している可能性のある端末を特定するというアプローチをとる．弁別については機械学習を利用する．本節では，識別器を訓練するためのデータと，性能を評価するためのデータの 2 種類について述べる．表 1 と表 2 にそれぞれ訓練データと評価データの内訳を示す．

2.1.1 訓練データ

悪性通信の訓練データとして，24,260 検体を動的解析することで得られた通信データを利用する．このデータは 2014 年 12 月から 2015 年 9 月までの間に検体を収集し，解析したものである．検体は Web 巡回型ハニーポッ

表 1　訓練データの内訳

traffic	samples	collection periods
malicious	24,260	2014/12-2015/09
legitimate	-	2016/06/25, 2016/06/29, 2016/07/07

表 2　評価データの内訳

traffic	samples	collection periods
malicious	15,000	2016/06/18-2016/06/24
legitimate	-	2016/06/19

表 3　各訓練データの TCP セッション数とフィールド数

traffic	TCP sessions	all fields	all fields(unique)
malicious	80,414	1,484,481	113,493
legitimate	81,310	929,327	131,586

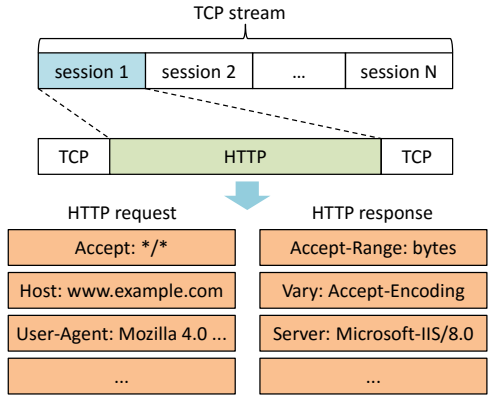

図 2　HTTP ヘッダフィールド抽出の概要

ト及び待受型ハニーポットによって収集した．各検体は TrendMicro [6] 及び Kaspersky [7] のいずれかのベンダが悪性であると判定したものであり，検体同士の重複は存在しない．

良性通信の訓練データとして，ある組織が使用しているネットワークの観測データを利用する．データは組織のゲートウェイにて収集したものであり，2016 年 6 月 25 日，29 日，7 月 7 日に tcpdump によってキャプチャした pcap 形式のデータである．また，URLBlacklist.com [8] のカテゴリ : spyware，virusinfected，warez のいずれかに登録されている URL にアクセスしているパケットはあらかじめ破棄しているが，それでも本ネットワークに接続している全ての端末がマルウェア感染端末ではないと断言はできない．しかしながら感染ホストが存在したとしても，それらの通信の割合は極めて小さいことが予想される．よって，本研究ではこのデータを良性通信データと定義する．

2.1.2　評価データ

悪性通信の評価データとして，15,000 種類のマルウェア検体の通信データを利用する．同一リソースによる検体の偏りを回避するため，Malwr [9]，MalShare [10]，VirusShare [11] の 3 ヶ所から，2016 年 6 月 18 日から 2016 年 6 月 24 日までの間にそれぞれ 5,000 検体ずつ収集した．検体は VirusTotal [12] によって少なくとも 1 つのベンダが悪性であると判定したものであり，訓練データと合わせて重複は存在しない．検体の通信データは，動的解析用のオープンソースソフトウェアである Cuckoo Sandbox [13] によって検体収集と同時期に取得した．

良性通信の評価データは，訓練データと同じ組織内の通信データを利用する．ただし，収集日は 2016 年 7 月 19 日であり，訓練データの収集時期とは異なる．このデータに関しても，訓練データと同様に BlackList による悪性通信の除去を行っている．

2.2　Step1: HTTP ヘッダフィールド抽出

Step1 では，HTTP ヘッダフィールドの抽出方法について述べる．近年のマルウェアは，インストール時や実行中，あるいは C&C サーバとの通信時に HTTP を利用するのが主流である [5]．そこで，本研究では HTTP のヘッダ内に記載されるフィールド情報に着目する．HTTP ヘッダフィールドの組み合わせは，ブラウザやソフトウェア，通信先のサーバの設定によって異なる．図 2 に本ステップの概要を示す．抽出する際に使用するツールとして，Wireshark の CLI 版である TShark [14] を使用した．TCP ストリームとは，全ての TCP セッションの集合であり，各 TCP セッションは主に 3-way handshake による接続確立及び切断確立の TCP パケットと HTTP パケットの集合で構成されている．HTTP ヘッダフィールドはコロンによって値が区切られており，前者が Key，後者が Value としてヘッダ内に記載されている．本ステップではこのフィールドの抽出を行うが，それと同時に URI 情報の path 部分と query 部分の抽出も行う．URI は厳密にはフィールドではないが，HTTP ヘッダに記載される情報という点では同一であるため，本研究ではこれも HTTP ヘッダフィールドの一種として扱う．また，URI には本来 key 部分は存在しないが，他のフィールドとの区別を容易にするために key 部分を「URI」とあらかじめ設定する．各 TCP セッション内の HTTP ヘッダフィールドを抽出する際に，以下に示す Key を持つフィールドに関しては Value を無視し，Key のみを抽出した．

- Accept-Language
- Date
- Expires
- If-Modified-Since
- Last-Modified

それぞれの Value は日付，あるいは端末の言語設定に依存する値であり，解析に用いた Sandbox の環境に強く依存するためである．また，プロキシサーバのようなミドルボックスを通過するパケットに付加される特殊なフィールドについては，Key と Value の値に関係なく抽出から除外する．これについても，特定の環境に強く依存することが理由に

表 4　HTTP ヘッダフィールドのテンプレート化例

HTTP header fields	HTTP templates
Accept: text json	Accept: text *
Accept-Encoding: gzip deflate	Accept-Encoding: gzip *
User-Agent: Mozilla 4.0 (compatible; MSIE 6.0; Windows NT 5.1)	User-Agent: Mozilla * (compatible; MSIE * Windows NT *)
Server: Apache 2.2.15 (Linux SUSE)	Server: Apache * (Linux SUSE)

挙げられる．表 3 に各訓練データから抽出した TCP セッション数と全フィールド数，ユニークなフィールド数を示す．良性通信よりも悪性通信のほうがフィールドの使用数は約 50 万程多い一方で，ユニークなフィールド数に関しては良性通信のほうが多いことがわかる．このことから，悪性通信ではある一定のフィールドが多用される傾向にあることが考えられる．

2.3 Step2: HTTP ヘッダフィールドのテンプレート化

抽出した HTTP ヘッダフィールドの数は非常に膨大である．それに加えて，一部の通信でしか使用されないフィールドも存在するため，抽出したフィールドをそのまま識別器で訓練すると，過学習を招く恐れがある．そこで，本研究では HTTP ヘッダフィールドを構成する各単語の可変性に着目し，テンプレート化による情報の圧縮を狙う．具体的な手法として，スコアリングと教師なし学習である DBSCAN（Density-Based Spatial Clustering of Applications with Noise） [15] を応用したアルゴリズムを組み合わせた独自のテンプレート化手法 [16] を適用する．DBSCAN とは，事前にクラスタ数を指定する必要がなく，任意形状にクラスタを抽出することができるアルゴリズムである．あらかじめ距離の閾値 Eps と対象数の閾値 $MinPts$ の 2 つを設定する．ある対象 p と q が存在するとき，ある点 p から距離 Eps 内にある点集合を，

$$N_{Eps}(p) = \{q \in D \mid dist(p,q) \leq Eps\} \quad (1)$$

と定義する．ここで以下の条件，

$$p \in N_{Eps}(q) \quad (2)$$

$$|N_{Eps}(q)| \geq MinPts \quad (3)$$

を満たす p と q を *directly density-reachable* と呼び，対象を同じクラスタに分類する．

本研究で用いるテンプレート化手法では，フィールド内の単語の名称と個数，そして各単語の出現位置に関する規則性に注目している．これは，フィールド内のパラメータとなりうる単語（バージョン情報やホスト名等）の出現頻度は低くなるという性質を利用している．例えば，User-Agent フィールドでは Web ブラウザのバージョンがフィールドのパラメータ部分に対応しているケースが多い．この場合，テンプレート化によってパラメータ部分をワイルドカードに置き換えることにより，同じブラウザの複数バージョンのフィールドを 1 つのテンプレートに圧縮することができる．これによって今後登場し得る未知のバージョンに関しても同じテンプレートで対応することが可能であるため，ロバスト性の確保にも貢献している．

2.3.1 単語のスコアリング

初めに，HTTP ヘッダフィールドを構成する各単語についてスコアリングを行う．ここでは，事前に各フィールドが Key 毎に保存されているものとする．それぞれのフィールドに対して，” 空白”，”/”，”=”，”,” の各文字列毎に分割を行う．これらの文字列を選択した理由は，これらの後にパラメータとなり得る可変性の高い単語が連続する確率が高いという経験則からである．URI に関しては，”?” や”&” がクエリや複数のクエリ間のセパレータとして機能しており，これらの文字列も分割の条件として適当であるが，URI 以外のフィールドの場合では余計な分割をしてしまう恐れがある．よって，この 2 種類の文字列に関しては分割は行わない．

次に，各単語の条件付き出現確率をスコアとして計算する．ここで，$n(X)$ はある変数値 X が出現する回数であるとする．計算式を以下に示す．

$$Score(word, pos, len) = P(word \mid pos,\ len)$$

$$= \frac{n(word\ \cap\ pos\ \cap\ len)}{n(pos\ \cap\ len)} \quad (4)$$

$word$ は単語の名称，pos はその単語が出現する位置，len はフィールド内の単語数を示す．可変のパラメータ部分のみが異なる各フィールドは $word$ 以外の条件が一致する．この構造により．パラメータ部分の単語のスコアが他の単語よりも相対的に低くなる．

2.3.2 クラスタリング

次に，各単語をクラスタリングすることで，パラメータとなりうる単語をノイズとして除去し，クラスタ内の単語を再構成することでテンプレートを作成する．あらかじめ閾値 δ（$\delta \geqq 0$）と閾値 β（$0 < \beta < 1$）を設定し，以下の手順でクラスタリングを行う．

単語のソート

単語のスコアの降順にフィールド内の単語をソートする．

クラスタの形成

スコアの順にクラスタを形成する．クラスタ内の平均スコアと次の単語のスコアの差が δ 未満である場合，現在のクラスタへ単語を追加する．そうでない場合は

次の単語を現在のクラスタとして設定する．フィールド内の全ての単語がクラスタに含まれるまで繰り返す．

テンプレート出力

クラスタを上から順に結合していく．クラスタ内の総単語数 $\geq len \times \beta$ となったとき，その時点でのクラスタ内の単語をテンプレートとして出力する．

表 4 に HTTP ヘッダフィールドのテンプレート化の例を示す．

2.4 Step3: 評価データの弁別と有効な特徴量の抽出

本ステップでは，評価データに対して機械学習を適用し，悪性通信と良性通信の弁別精度の評価を行う．機械学習により，未知のマルウェアにも対応可能なヒューリスティックな検知を実現する．識別に用いる特徴として，前ステップで生成した HTTP テンプレートを重複を除いて使用する．評価データは，TCP セッション毎に重複を除いた HTTP ヘッダフィールドをベクトル化して識別器に入力する．学習モデルは L1 正則化スパース学習アルゴリズム [17] を採用した．スパース学習は，弁別に貢献しない特徴を極力削除しながら識別器の最適化を図る学習アルゴリズムである．これにより，特徴量として有効に機能したテンプレートを抽出することができる．詳しくは 3 章にて述べる．識別器のアルゴリズムはロジスティック回帰を採用した．特徴として使用するテンプレートがセッション中で使われているか，使われていないかを $x \in \{0, 1\}$ の二値変数で表現する点，そして特徴として利用するテンプレートの数が膨大であるという点から妥当な選択である．スパース学習を行うための最適化にあたって，L1 正則化（Lasso）項をつける．最終的な最適化問題は以下の式で与えられる．

$$\min_{w} ||w||_1 + C \sum_{i=1}^{l} \log(1 + e^{-y_i w^T x_i}) \qquad (5)$$

ここに (x_i, y_i) は訓練データ（x_i が i 番目のデータの特徴，y_i が i 番目のデータのクラス），w はロジスティック回帰のパラメタ，C は二乗誤差と L1 拘束条件のバランスを調整するパラメタ，$||\cdot||_1$ は L1 ノルムである．最適化を解く実装として，オープンソースの機械学習ライブラリである LIBLINEAR [18] を使用した．

3. 実験結果

本章では，テンプレート化の効果とスパース学習による弁別精度，そして有効な HTTP テンプレートの抽出結果を述べる．図 4 に閾値の違いによるテンプレート数の変化を示す．$\delta = 1.0$ は，テンプレート化を行わなかった場合の結果である．δ 及び β が小さい程，テンプレート化の効果が大きくなることがわかるが，特に $\beta = 0.5$ の際のテンプレート数の減少が著しい．ここで，$\beta = 0.55$，フィールド内の元の単語数:*len* が 2 であるフィールドのテンプレート化について考察する．クラスタリングアルゴリズムより $0.55 * 2 = 1.1$ となり，クラスタ内の単語数が 1.1 以上の整数の中で最小の値である 2 となったときに，そのクラスタをテンプレートとして生成する．よって，*len* が 2 であるフィールドはテンプレート化が行われない．ここで，図 3 に *len* の累積分布を示す．右の拡大図に注目すると，*len* が 2 であるフィールドが約 50 %を占めており，支配的であることがわかる．$\beta = 0.55$ ではこれらのフィールドはテンプレート化されないが，$\beta = 0.5$ のときは $0.5 * 2 = 1.0$ となり，*len* が 2 であるフィールドのテンプレート化も可能となる．また，$\beta = 0.45$ 及び $\beta = 0.4$ のテンプレート数を見ると，$\beta = 0.5$ と比べてほとんど変化はない．これらの結果から，我々は $\beta = 0.5$ が最適な閾値であると判断した．表 5，6，7，8 に閾値 δ とスパース学習のコストパラメータ C の値をそれぞれ可変させた際の結果を示す．結果では ACC，FPR，FNR を算出する．それぞれは以下の式で計算される．

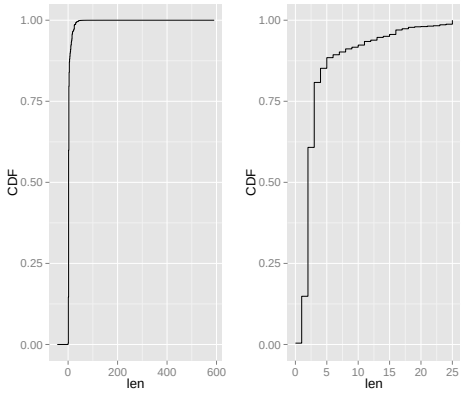

図 3 フィールド内の単語数:*len* の累積分布
（左：原寸図　右：拡大図）

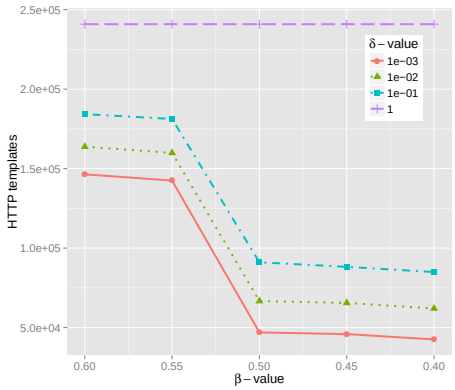

図 4 閾値の違いによるテンプレート数の変化

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \qquad (6)$$

表 5　$\delta = 10^0$（テンプレート化無し）

C	features	ACC	FPR	FNR
10^{-3}	28	0.7329	0.2030	0.3348
10^{-2}	94	0.7448	0.07385	0.4470
10^{-1}	367	0.7658	0.01038	0.4710
10^{0}	1,683	0.7873	0.006782	0.4306
10^{1}	8,503	0.7456	0.009738	0.5133
10^{2}	48,775	0.8028	0.01137	0.3939

表 6　$\delta = 10^{-1}$

C	features	ACC	FPR	FNR
10^{-3}	25	0.7192	0.1004	0.4716
10^{-2}	110	0.7509	0.03611	0.4745
10^{-1}	348	0.8029	0.01459	0.3902
10^{0}	1,265	0.8734	0.008791	0.2512
10^{1}	5,502	0.9046	0.01641	0.1789
10^{2}	23,306	0.8856	0.03535	0.1979

表 7　$\delta = 10^{-2}$

C	features	ACC	FPR	FNR
10^{-3}	27	0.7090	0.1018	0.4912
10^{-2}	100	0.7375	0.03644	0.5017
10^{-1}	315	0.7944	0.01497	0.4073
10^{0}	1,192	0.8342	0.01197	0.3285
10^{1}	5,114	0.8733	0.01508	0.2449
10^{2}	20,969	0.8032	0.02565	0.3780

表 8　$\delta = 10^{-3}$

C	features	ACC	FPR	FNR
10^{-3}	22	0.7098	0.1015	0.4899
10^{-2}	97	0.7376	0.03656	0.5012
10^{-1}	319	0.8061	0.01406	0.3843
10^{0}	1,191	0.8120	0.009321	0.3770
10^{1}	4,776	0.8892	0.01425	0.2130
10^{2}	14,487	0.8750	0.02050	0.2356

$$FPR = \frac{FP}{FP + TN} \tag{7}$$

$$FNR = \frac{FN}{FN + TP} \tag{8}$$

TP は，悪性であると判定した TCP セッションが実際に悪性である回数であり，TN は良性であると判定したセッションが実際に良性である回数を示す．FP は悪性であると判定したセッションが実際は良性だった回数であり，FN は良性であると判定したセッションが実際は悪性だった回数である．一般に ACC は高い程，FPR 及び FNR は低い程良いとされる．表 5 のテンプレート化を施さなかった場合の ACC は 80%を前後している一方，テンプレート化を施した際の ACC はどれも精度が向上している．前者の ACC の最高値は $C = 10^2$ のときの 80.3%であり，FPR の最低値は $C = 10^0$ のときの 0.678%であった．一方で後者の ACC は $\delta = 10^{-1}$，$C = 10^1$ のときに最大値 90.5%を記録し，FPR は $\delta = 10^{-1}$，$C = 10^0$ のときに最低値 0.879%を記録した．精度に関してはテンプレート化によって約 10%向上することを示した．これは，テンプレート化による情報の圧縮と過学習の回避が，精度の向上に貢献していると考えられる．FPR については僅かながら上昇してしまったが，どちらも 1%未満を維持しており，大差はない．

次に，ACC が最大となった $\delta = 10^{-1}$，$C = 10^1$ のときに，5,502 種類の弁別に有効な特徴を抽出することができた．図 5 にテンプレートを Key 毎に集計した際の，特徴量を占める割合の上位 10 種類を示す．URI の割合が最も多く，特徴量の 40 %以上を占めることがわかる．さらに，上位 2 種類の Key である URI 及び User-Agent フィールドは実際に先行研究 [19]，[20] でも活用されており，有効な特徴量である．表 9，10 に各通信を弁別する際に貢献した具体的なテンプレートの上位 10 種類を示す．テンプレートが長過ぎるものに関しては上位 80byte 以降を省略して記載する．重みが正の値であるものは悪性通信を弁別する際に貢献したものであり，負の値であるものは良性通信を弁別する際に貢献したものである．ここでも，URI と User-Agent が特徴として貢献していることがわかる．スパース学習によって抽出した特徴量は，全てが普遍的な特徴といえるわけではない．例えば良性通信の重みの上位に Cookie フィールドがいくつか確認できるが，Cookie フィールドはセッション毎にランダムな値が利用されるため，これは局所的な特徴である．

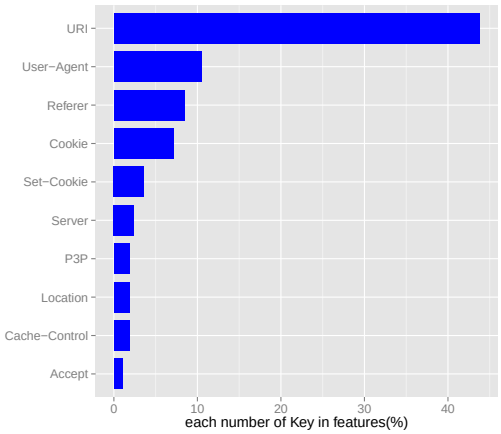

図 5　各 Key の特徴量内の割合-Top10

4. 関連研究

関連研究では TCP フィンガープリントを利用するもの，User-Agent に着目したもの，そして HTTP ヘッダに着目したものがある．ここではそれらの概要を述べ，本研究との差異を述べる．

表 9　悪性通信の弁別に貢献した特徴量（HTTP テンプレート）-Top10

features	weight
X-RTFM: Learn about this site at http: bit.ly icanhazip-faq and don't abuse the	10.86
URI: test.html	9.974
URI: wapi event raise	9.622
URI: dm.php?uid 20&tid 1&ref 1	9.533
URI: Data.txt	8.797
User-Agent: Mozilla 5.0 (Windows NT 5.1; rv:36.0) Gecko 20100101 Firefox 36.0	8.740
URI: new rar.xml	8.444
User-Agent: Proxy 1.0	8.408
Server: thin 1.4.1 codename Chromeo	8.384
Accept-Encoding: gzip deflate gzip deflate	8.242

表 10　良性通信の弁別に貢献した特徴量（HTTP テンプレート）-Top10

features	weight
URI: get ?ver 2.3.0.3445&aid 848366190&hid e5cc5079164c1dfd&rid 1467883972088&da	-11.78
URI: OnlineRegister MemberOnRegCheck.aspx?asn B5B617028D19B7EDFB1B3231ED347D00&	-9.602
Cookie: BDUSS Ntfn5aeXVSMm9zVjZxcTM2bmlURWVaTVZCVHhRWTU3ckVRMEtkflRyQXY2VVpYQVF	-9.072
srvtag: *	-8.180
User-Agent: Microsoft-CryptoAPI 10.0	-8.133
User-Agent: Mozilla 4.0 (compatible; MSIE 6.0; Windows NT 5.1)	-7.777
URI: imageproxy2 dimgm scaleImage?url http%3A%2F%2Fimg.kxting.cn%2F%2Fdmimg11%2	-7.553
Cookie: IPLOC unknown; YYID 16671165628453679CAB1A9508DED6C0; IMEVER 8.0.0.8083	-7.473
Cookie: YYID DF4C3586051924A42AAE8F4ED36483A2; CXID 5498024289BFC19E259DB110C75	-7.466
URI: wangwang get_task.php?id 99999&vr 1.650000&th 1&it 60&pv	-7.393

4.1 TCP フィンガープリント

木佐森ら [21] は IP ヘッダと TCP ヘッダの情報を利用した OS フィンガープリンティングにより，カーネルマルウェアによる悪性通信が存在する可能性を指摘している．カーネルマルウェアは最も権限の高いレベルで動作するマルウェアであり，メモリや CPU 命令等にフルアクセスが可能である．そのため，通信は OS に依存しない独自のフィンガープリントを持つと分析されている．しかしながら通常のマルウェアによる通信は OS に依存しているため，OS フィンガープリンティングのみでは通常のマルウェアによる悪性通信を特定することは困難である．

4.2 User-Agent

Zhang ら [22] は正規表現による検知システムを回避する User-Agent の存在を明らかにし，WEB ブラウザが利用する User-Agent の属性を細かく分析している．また，この情報と OS フィンガープリンティングを活用することで偽造された User-Agent が特定可能であることを示している．User-Agent の情報は弁別において非常に有効だが，それだけでは十分とは言えない．

Grill ら [23] は User-Agent には *Legitimate user's browser*, *Empty*, *Specific*, *Spoofed*, *Discrepant* の 5 つのパターンが存在するという．それぞれの意味は，『正規の User-Agent』，『表記がない User-Agent』，『ブラウザによるものではない User-Agent』，『マルウェアによるホストが使用しているブラウザを模した User-Agent』，『マルウェアによる設定が矛盾している User-Agent』を指す．Spoofed の場合はユーザ個人が使用しているブラウザの User-Agent をそのまま利用する場合が多いため，それ単体の情報のみでは検知することは困難であるとしている．しかし 3 つの検知アルゴリズムを用いることで，Spoofed 以外を『矛盾した User-Agent』，『マルウェア独自の User-Agent』，『表記されていない User-Agent』の 3 つに分類し検知することが可能であるとしている．この研究でも User-Agent の情報しか使用しておらず，やはり十分とは言えない．

4.3 HTTP ヘッダ

Chiba ら [19] は正規表現によるテンプレートを利用した検知手法を提案している．まず，HTTP リクエストパケットのフィールドの URL パス，URL クエリ，そして User-Agent の可変性をプロファイリングし，正規表現化を行い抽出をする．次に一段回目に IP アドレスによるクラスタリングを行い，二段階目にあらかじめ定義した計算方法を用いて類似度を計算し，それに基づいて階層的クラスタリングを行う．これによりテンプレートを生成し，マッチングにより悪性通信を検知する．この手法により高い精度を維持しつつ誤検知率も下げることに成功している．しかしながらテンプレートマッチングの場合，未知の悪性通信に対しての対応に不安が残る．

5. まとめ

5.1 結論

本研究では，24,260 種類のマルウェア検体通信データと，同じ量に匹敵する良性通信データの各 HTTP ヘッダ情報

を利用することで，インターネット・ゲートウェイからそれぞれの通信を高精度に弁別する軽量なシステムを提案した．このシステムは，ヘッダ内に記載される HTTP ヘッダフィールドの組み合わせと可変性に着目することで，悪性通信の高精度な検知を可能としている．テンプレート化技術とスパース学習アルゴリズムを用いることで，特徴量の削減と精度の大幅な向上を定量的に示した．さらに，特徴量として弁別に貢献したテンプレートを抽出し，その傾向を明らかにした．実データを利用した実験では，全く別の環境・時期に収集した 15,000 種類のマルウェア検体の悪性通信と同規模の良性通信を利用することで，このシステムが有効に機能するかを評価した．その結果，誤検知率を低く維持しつつ，90.5%という高精度で検知が可能であることを示した．

5.2 今後の課題

本研究では，FNR の値が ACC や FPR の値と比べて相対的に高くなっている．マルウェアにはインストール時，あるいは実行中に，感染している端末内の Web ブラウザを起動させ，外部と通信を行う検体が存在する．本研究では良性通信データにブラウザ通信が多く含まれているため，ブラウザを介した悪性通信が行われる場合，良性通信であると誤判定する可能性がある．今後はこの問題を検討する必要がある．

スパース学習による特徴量抽出にも課題が残っている．結果では Cookie 及び Set-Cookie フィールドの割合が上位に挙がっているが，これらはセッションによって Value が異なるため，汎用性があるとはいえない．つまり，スパース学習は必ずしも普遍性の高い特徴のみを抽出するわけではない．よって，スパース学習以外の次元削減アルゴリズムを採用することも視野にいれるべきである．

本システムは検体の HTTP ヘッダ情報のみを利用するため，他のプロトコルを利用するマルウェアには対応することができない．しかし，そのようなマルウェアに対しても各プロトコルのヘッダ情報を利用することで，本研究の手法を適用できる可能性はある．

謝辞 本研究を進めるにあたり，テンプレート化に関する技術を提供して頂いた NTT ネットワーク基盤技術研究所の木村達明氏に感謝いたします．

本研究の一部は JSPS 科研費 JP16H02832 の助成を受けたものです．

参考文献

[1] “Quarterly Threat Report: What Do the Numbers Mean to Me?.” https://blogs.mcafee.com/consumer/quarterly-threat-report-numbers-mean/.

[2] “Market Share Statistics for Internet Technologies.” https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=0.

[3] “ESET predictions and trends for cybercrime in 2016.” http://www.welivesecurity.com/2015/12/23/eset-predictions-for-cybercrime-trends-in-2016/.

[4] “Large CCTV Botnet Leveraged in DDoS Attacks.” https://blog.sucuri.net/2016/06/large-cctv-botnet-leveraged-ddos-attacks.html.

[5] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy, “Studying spamming botnets using botlab.,” in *NSDI*, vol. 9, pp. 291–306, 2009.

[6] “Content security software - Internet Security & Cloud - Trend Micro USA.” http://www.trendmicro.com.

[7] “Kaspersky Lab |Antivirus Protection & Internet Security Software.” http://www.kaspersky.com/.

[8] “URLBlacklist.com.” http://urlblacklist.com/.

[9] “Malwr - Malware Analysis by Cuckoo Sandbox.” https://malwr.com/.

[10] “MalShare.” http://malshare.com/.

[11] “VirusShare.com.” https://virusshare.com/.

[12] “VirusTotal - Free Online Virus, Malware and URL Scanner.” https://www.virustotal.com/.

[13] “Cuckoo Sandbox: Automated Malware Analysis.” https://www.cuckoosandbox.org/.

[14] “tshark - Dump and analyze network traffic.” https://www.wireshark.org/docs/man-pages/tshark.html.

[15] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.,” in *Kdd*, vol. 96, pp. 226–231, 1996.

[16] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto, “Spatio-temporal factorization of log data for understanding network events,” in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 610–618, IEEE, 2014.

[17] R. Tibshirani, “Regression shrinkage and selection via the lasso,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[18] “LIBLINEAR – A Library for Large Linear Classification.” https://www.csie.ntu.edu.tw/~cjlin/liblinear/.

[19] D. Chiba, T. Yagi, M. Akiyama, K. Aoki, T. Hariu, and S. Goto, “Botprofiler: Profiling variability of substrings in http requests to detect malware-infected hosts,” in *Trustcom/BigDataSE/ISPA, 2015 IEEE*, vol. 1, pp. 758–765, IEEE, 2015.

[20] T. Nelms, R. Perdisci, and M. Ahamad, “Execscent: Mining for new c&c domains in live networks with adaptive control protocol templates,” in *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pp. 589–604, 2013.

[21] 木佐森幸太, 下田晃弘, 森達哉, and 後藤滋樹, “TCP フィンガープリントによる悪意のある通信の分析,” **情報処理学会論文誌**, vol. 52, no. 6, pp. 2009–2018, 2011.

[22] Y. Zhang, H. Mekky, Z.-L. Zhang, R. Torres, S.-J. Lee, A. Tongaonkar, and M. Mellia, “Detecting malicious activities with user-agent-based profiles,” *International Journal of Network Management*, vol. 25, no. 5, pp. 306–319, 2015.

[23] M. Grill and M. Rehak, “Malware detection using http user-agent discrepancy identification,” in *Parallel Computing Technologies (PARCOMPTECH), 2015 National Conference on*, pp. 221–226, IEEE, 2015.