Computer Security Symposium 2016
11 - 13 October 2016

データ圧縮アルゴリズムを用いたマルウェア感染通信ログの判定

岡野 靖　神谷 和憲　谷川 真樹[†1]

概要：近年、端末のマルウェア感染についての通信ログ判定手法に機械学習技術が盛んに適用されつつある。機械学習においては対象データからの特徴抽出手法もその判定精度に大きな影響を与えるが、通信ログにおいてはしばしば未知のデータ列を含むため、通常の特徴抽出手法では適さない場合がある。本研究では通信ログからの特徴抽出として、gzip や compress 等で用いられるデータ圧縮アルゴリズムを用いた手法の適用を試みた。本手法は特にマルウェアのパターン化された通信先 URL 等で良く特徴を捉え、通常の特徴抽出手法よりも良い判定精度を得ることができた。

キーワード：マルウェア、通信ログ、データ圧縮、機械学習

Log Detection of Malware-infected Host Communications with Data Compression Algorithm

Yasushi Okano　Kazunori Kamiya　Masaki Tanikawa[†1]

Abstract: Recently, machine learning is actively applied to log detection of Malware-infested Host Communications. Feature extraction in machine learning is a significant impact on the detection accuracy. But conventional feature extraction methods (ex. bag of words) is not be suitable for communication log, because it is often containing unknown data sequence. We propose new feature extraction method from the communication log using a data compression algorithm, gzip, compress, and so on. This method has especially caught patterned URL or the like malware, it is possible to obtain a good detection accuracy than conventional feature extraction methods.

Keywords: malware, log, data compression, machine learning

1. はじめに

近年、亜種・新種のマルウェアがより多く出現するに伴い、アンチウイルスソフト等での検知をすり抜け，マルウェアに感染する端末が出現する可能性を十分に考慮して対策を行う必要がある．これら感染端末の検知の一手法として，ファイアウォールやプロキシサーバ等のネットワークの出口の装置にて記録される通信ログを分析し判定する手法がある．通信ログに対して，通信先 FQDN や URL をブラックリストと突合したり，マルウェア感染時の通信の振る舞いをルールとして規定し，そのルールに合致するかを探索したりする等の分析・判定がよく行われている．

一方，コンピュータ性能の高まりやマルウェア検体等の過去のデータの十分な蓄積に伴い，機械学習の手法を用いて，より効率的かつ高精度な通信ログの判定・分析を可能とする研究が盛んに提案されつつある．一般的な機械学習手法では，対象データから注目すべきいくつかの情報を特徴として抽出，数値化し，その数値データである特徴ベクトルを機械学習アルゴリズムにより学習・推論することが行われる．そのため，機械学習アルゴリズムのみならず，特徴抽出手法もその性能を作用する重要な要素である．

本研究は，通信ログにおけるマルウェア感染端末判定において，特に重要な情報源となる通信先 URL からの特徴抽出手法として，データ圧縮アルゴリズムを用い，その圧縮率を特徴として用いる手法を提案する．また，本特徴抽出手法を教師あり学習に適用することで，従来の特徴抽出手法を用いたマルウェア感染端末判定よりも高精度の判定をおこなえた事を報告する．

†1 日本電信電話株式会社
Nihon Telegraph and Telephone Corporate

2. 関連文献・研究

2.1 データ圧縮アルゴリズムを用いた分類

Benedetto らは情報エントロピーや Kolmogorov 複雑性等の情報量の指標のひとつとして，データ圧縮アルゴリズムを用いた指標である関連エントロピー(relative entropy)を提案した[1]．同じコードの連続やコードパターンの繰り返し等，何らかの類似パターンが出現するデータをデータ圧縮アルゴリズムは良く圧縮することに注目すると，データ A から見たデータ x の情報量の多さはその圧縮されにくさに関連すると思われる．そこで，以下のようにデータ圧縮を用いて，データ A と x の関係エントロピー$C_A(x)$を定義した；

$$C_A(x) = Z(A\ cat\ x) - Z(A)$$

ただし，Z はデータ圧縮後のサイズを出力する関数，cat は前者の最後尾に後者をデータ結合する操作である．または，上記をデータ x のサイズで割り，正規化したものを関

係エントロピーとして用いる場合もある．

この提案は分類問題への応用が試みられた[2][3][4][5][6]．推論データ x が分類 A,B のいずれかであることを推論するのに，x を各分類の訓練データ A,B のより類似している方に分類するという考えは自然である．類似は情報量の少なさ，すなわち，関係エントロピーの少なさで測定でき，より関係エントロピーが小さい方に推論データは分類されうる．文献[4]はSPAMフィルタに適用し，従来のBag of Wordsによる特徴抽出と教師あり分類器による手法よりもよい精度が得られたと報告している．文献[5]は，Twitter サービスの短文メッセージ（Tweet）の分類問題に対し，以下のようにスムージングパラメータγを導入した手法を適用した；

$$\text{Score} = \frac{C_A(x)+\gamma}{C_B(x)+\gamma}$$

Score が小の場合に分類 A，大の場合は分類 B と推論される．γを大きくすると，相対的に文字数が少ないメッセージの影響を少なくすることができ，より意味がある比較的長文のメッセージに注目することができるように調整することが可能である．上記手法が，特徴抽出に形態素解析，分類アルゴリズムに CW 法を用いた機械学習手法よりもよりよい分類精度を得られたことを報告している．

データ圧縮アルゴリズムは，テキストデータに対して LZSS (gzip), LZW (compress), PMP (rar) が利用できるとの報告されている[2][3][4][5]．一方，楽譜から作曲者を判定する手法においては bizp2 が有効であるとの報告[6]がある．

2.2 ログ，特に URL に関する特徴抽出手法

テキストの特徴抽出手法として Bag of Words (BoW)の手法が良く知られている．これは，適切な区切り文字や形態素解析等で，テキストを語として分解し，語 1 つ 1 つを特徴ベクトルの 1 つの次元とし，語の出現数等適切な重みで数値化を行うものである．URL や User-Agent 等のテキストにしばしば適用される．

一方で，セキュリティ知識を生かし，たとえば，通信先 URL 中にダッシュ‘ー’が多いものはマルウェア等，ヒューリスティックな手法を特徴抽出として用いるものが提案されている．

文献[7]は，URL をいったん正規表現等のパターンに変換した後にクラスタリングを実施しており，パターン化により，類似した特徴同士を 1 つに纏め上げることに成功している．

3. 提案手法

ログのアクセス先 URL に対してデータ圧縮アルゴリムを用いた特徴抽出を行い，その特徴ベクトルを用いて教師あり機械学習による分類手法を提案する．圧縮アルゴリズムを用いることにより，URL 文字列中の同一パターンをデータ圧縮アルゴリズムが自動的に効率よく認識する可能性がある．

従来の圧縮による分類手法は，SPAM メールのテキストや Tweet 等，1 データすべてを 1 つの属性と見て，データ圧縮アルゴリズムによる分類を行っていた．マルウェアの通信先 URL は，同一種の C&C サーバが複数あり，状況に応じてその通信先を変更する等の事例が見られることから，少なくとも FQDN と Path 以降の 2 属性に分けて判定したほうがより良く判定できることが期待される．本方式では複数の属性に対応可能とするため，従来のようにデータ圧縮アルゴリズムで直接分類するのではなく，関係エントロピーを用いた特徴として抽出し，その特徴を改めて教師あり分類器を用いて分類する手法を提案する．以下に本手法の手順を列挙する；

1. 訓練データ (悪性ログ，良性ログ) を事例選択する．
2. 事例選択済み訓練データの指定の属性をデータ圧縮特徴抽出器へ登録する
3. 登録済みデータ圧縮特徴抽出器を用いて訓練データを特徴抽出し，属性ごとに悪性圧縮率(Zpos)，良性圧縮率(Zneg)を特徴ベクトルとして得る
4. 上記特徴ベクトルに変換された訓練データで，悪性ログに含まれるデータを悪性，良性ログに含まれるデータを良性としてクラス付けし，教師あり分類器を訓練する
5. 推論時，登録済みデータ圧縮特徴抽出器を用いて推論データ(判定対象ログ)より特徴抽出し，訓練済み分類器にて推論し，スコアを得る．
6. 判定閾値以上のスコアとなったログの発信元端末を感染端末と判定する．

以下，特にデータ圧縮特徴抽出に関連する手順 1-3 について加えて説明する．

1 の事例選択は，感染端末通信にもしばしば存在する，非感染端末でもよくアクセスされる URL を悪性ログから削減等を行う．マルウェア OutBrowse の通信先 URL を時系列順に表 1 に示す，4 行目は Google 社が提供する API の URL であり，非感染端末からも利用される URL である．これら良性・悪性共に出現する URL はデータ圧縮での分類にて著しく判定精度を悪化させる．このことを考慮し，a) 良性ログにも出現する URL を悪性ログより除去，b) 各ログ内で重複するログを 1 つのみ残して削減，の 2 つの事例選択を施す．

表 1 マルウェア OutBrowse の通信先 URL

1:	http://installer.apps-track.com/Installer/Flow?pubid=7302&distid=13467&productid=1694&subpubid=-1&campaignid=0&networkid=0&dfb=-1&os=5.1&iev=8.0&ffv=&chromev=&macaddress=&netv=&d1=15693&d2=-1&d3=-1&d4=-1&d5=-1&ds1=&hb=2&systembit=32&vm=1&version=4.0
2:	http://serv.the-app-data.info//offers/DynamicOfferScreen?offerid=2&distid=13467&leadp=1694&countryid=142&sysbit=32&dfb=-1&hb=2&isagg=0&version=4.0&external=0&external=0&
3:	http://cdn.file7desktop.com/ProductS/PlayerStubWrapper1.exe
4:	http://ajax.googleapis.com/ajax/libs/jquery/1.5/jquery.min.js …(中略)…
5:	http://d.castplatform.com/api/vp/1?clk=gLg_PHWA9aSyioXkt-F4b3J9cI1ybf-t-x7VxWH5dmAWXwln-z (以下略)

2 の訓練データのデータ圧縮特徴抽出器への登録方法は用いるデータ圧縮アルゴリズムにより異なる．次節の実験方法にて説明する．なお，訓練データの形式は URL 属性なら，表 1 のような URL のリストを良性，悪性ごとに取りまとめたものとなる. FQDN 属性や Path 属性でも同様に（事例選択した）良性，悪性ごとの FQDN や Path のリストをそのまま訓練データとする．

3 のデータ x の悪性圧縮率 Zpos，良性圧縮率 Zneg の定義は以下のとおりである;

$$Z_{pos}(x) = \frac{C_{pos}(x) + \gamma}{L(x) + \gamma}$$

$$Z_{neg}(x) = \frac{C_{neg}(x) + \gamma}{L(x) + \gamma}$$

ただし，$C_{pos}(x)$，$C_{neg}(x)$ はそれぞれ悪性ログおよび良性ログとデータ x の関係エントロピー，$L(x)$は x のデータサイズ，γはスムージングパラメータを示す．訓練データは手順 1 で事例選択を行ったデータを用いてもよいし，事例選択を行う前のデータを用いてもよい．今回は事例選択を行う前のデータを用いた．

4. 実験方法

本提案の方式において，どのようなデータ圧縮アルゴリズムおよび教師あり分類器が適しているのかは, BoW のような従来方式との比較とともに興味を引くところである．実験ではまず，判定精度および速度性能の面からどのデータ圧縮アルゴリズムが本方式に適するかを明らかにした．次にその適したデータ圧縮アルゴリズムを用いて，分類器および特徴抽出に用いる属性を変えて（属性選択），感染端末の判定精度を評価した．また，ブラックリストを用いた判定，および，URL の特徴抽出に BoW を用いたサポートベクタマシン（SVM）での判定精度を対照実験として行った．

判定精度の検証は時系列ホールドアウト検証（フォワードテスト）[8]を行い，検証用データは訓練データよりも時間的に後のものを用いた．判定精度の指標として，AUC（Area Under the Curve），pAUC（partial AUC）[9]，および，誤判定率 (False Positive Rate) FPR＜0.5%となるように判定閾値を調整した際の判定率 (True Positive Rate) TPR である $TPR_{0.5\%}$を用いた[8]．AUC は判定閾値を変化させたときに FPR を横軸に，TPR を縦軸にとったときの曲線下の面積，pAUC は特定の FPR の範囲[p1,p2]での曲面下の面積として与えられる．TPR を FPR で定まる関数と見るとき，AUC, pAUC は以下で定義される；

$$\mathrm{AUC} = \int_0^1 TPR\ dFPR$$

$$\mathrm{pAUC} = \int_{p_1}^{p_2} TPR\ dFPR$$

pAUC は [p1, p2] = [0, 0.1] で算出されることが多く，本研究でもそれに倣った．感染端末は非感染端末よりもごく少数であることから，誤判定が少ないことがより重要であり，pAUC，および，$TPR_{0.5\%}$は低誤判定率時の判定率を評価する指標として扱うことができる．

データには，マルウェア検体の動的解析により収集した通信ログを悪性ログとし，社内ネットワークのプロキシログを良性ログと見なしたものを用いた．本実験で利用したログの諸元は以下のとおりである．なお，悪性ログについては，1 検体=1 端末として評価した．

悪性ログ：2015 年 2～7 月採取のマルウェア検体より
検体数(端末数) 71,310 件，ログ件数 7,152,479 件
良性ログ：2014 年 2～3 月採取の社内 NW プロキシログ
端末数 1,940 件，ログ件数 36,581,398 件

4.1 データ圧縮アルゴリズムの選定

特徴選択に用いるデータ圧縮アルゴリズムとして, LZSS (zip, LZ77), LZT (compress の圧縮アルゴリズム LZW, LZ78 の亜種) [10], bzip2, LZMA の 4 つを検証した．検証は時系列ホールドアウト検証で行い，悪性ログと良性ログの前半分のそれぞれ先頭 10 万件を訓練データ, 同様に後半分のそれぞれ先頭 10 万件を検証データとした．実施手順として，属性は URL を用いて，提案方式の 1-3 までを行い，良性・悪性圧縮率から以下の式でスコアを算出し，このまま，ログ行単位で指標を算出，評価した．

$$\mathrm{Score} = Z_{neg}/Z_{pos}$$

事例選択済み訓練データの登録については各データ圧縮アルゴリズムの特性に応じて，以下のとおり実施した．

LZSS はスライド辞書方式による圧縮を行うアルゴリズムである．圧縮対象のデータ列はその直前のスライド窓の領域のデータの中から最長一致するデータ部分を検索し，その相対位置とサイズのみを記録することで，データ圧縮を行う．そのため，スライド窓からはみ出した前方部分の情報は圧縮に加味されないことになる．一般的な LZSS のスライド窓は 32kB である. 本実験では, 訓練データを 20kB に分割して登録した．対象のデータ x の圧縮率を算出する際は，登録された各分割訓練データそれぞれにデータ x を結合，LZSS 圧縮し，もっとも圧縮率が小さいものを採用することとした．これにより圧縮に加味されない訓練データをなくすことができる.

LZT は compress や gif 形式ファイルで用いられているデータ圧縮アルゴリズム LZW の亜種であり，LZW と同様に動的辞書方式による圧縮を行うアルゴリズムである. 逐次，新規のデータ列があれば，辞書に追加していき，圧縮対象のデータは辞書で最長一致したデータ列の辞書 No.で代替されることで，データ圧縮を行う．辞書は Trie 木で作成される. 本実験では, 訓練データを Trie 木の辞書に登録する．データ x の圧縮率は，この辞書のみを参照して x をデータ圧縮し，算出することとした．辞書 No.の bit 長が辞書の大きさを決め，本実験は 24bit 長を用いた．LZW と LZT の違いは，前者が辞書満杯となった場合，以降の登録作業を中止してしまうのに対し，後者は利用されていないデータ列を辞書から破棄（LRU: Least Recently Used）して，新規データ列の登録を続行する点である.

bzip2，LZMA は，前者はブロックソート，後者は Markov アルゴリズムを用いてデータ圧縮を行う. LZSS と同様に，bizip2 はブロック領域，LZMA はスライド辞書領域を必要とするが，その領域は LZSS よりも十分大きい．そこで本実験では訓練データの登録時はそのままこのデータを保持することとした．データ x の圧縮率は，単純にこの訓練データと x を結合，圧縮することで求める.

LZSS，bzip2，LZMA は Python よりそれぞれ libz，libzip2，liblzma を利用することで実施した．LZT は Cython を用いて実装した.

スムージングパラメータはすべて γ=10 固定とした.

4.2 分類器および属性選択の比較

分類器はもともとのデータ圧縮分類の定義（たとえば[5]）により，線形でも十分に分類できると予想されるため，線形分類器を中心に選び，以下の 5 種類を比較検討した;
Ridge 回帰（Ridge）, L2 正則化サポートベクタマシン（SVM）, Fischer の線形判別法（LDA），ナイーブベイズ(NB)，AdaBoost.
分類器は Scikit-Learn のライブラリを使用した.

属性は URL，および，URL を FQDN と Path 以下の部分（Path と以後呼称）に分けたものとし，URL，FQDN，Path の 3 属性から特徴抽出を行い，その属性の組み合わせ（属性選択）により，判定率の違いを検証した.

対照実験としてブラックリスト判定，および，BoW 特徴抽出による L2 正則化 SVM での判定を実施した.

ブラックリストは, 悪性ログの通信先 URL を FQDN 部，PATH 部，QUERY 部にわけてそれぞれブラックリストとして保持した．QUERY 部は Value 部分の値をマスクした．さらに良性データとブラックリストを突合し，合致したものはブラックリストから除いた．こうして作成した 3 つのブラックリストのどれか1つにでも送信先 URL が合致した通信ログの送信元ホストはマルウェア感染ホストと判定した.

BoW 特徴抽出では, /, ?, &, = の 4 文字を区切り文字として，単語を抽出した.

検証は時系列ホールドアウト検証で行い，悪性ログと良性ログの前半分すべてを訓練データ，同様に後半分すべてを検証データとした．実施手順として，データ圧縮を行うものは提案方式の 1-6 を行い，スコアは同一送信元ホスト中でもっとも高いスコアをそのホストのスコアとし，ホスト単位での AUC，pAUC 等の指標を算出した.

スムージングパラメータはすべて γ=10 固定とした.

5. 実験結果

5.1 データ圧縮アルゴリズムの選定

データ圧縮アルゴリズム別に測定した判定率を表 2 に示す．これらの指標はログ単位で測定した指標であり，ホスト単位での指標より，大きい（良い）値が出るので注意する.

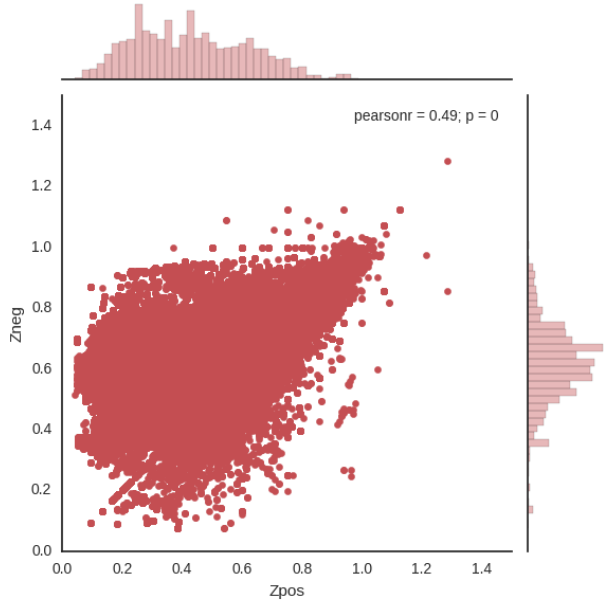
もっとも判定精度が良いのは LZMA であるが, もっとも実行時間がかかっており，1 千万件以上のログ分析に供するには実行時間の面で困難が予想される．LZT は高速であり，判定精度も優れることから，後の実験の特徴抽出には LZT を用いることとした．bizp2 はまったく判定できなかったといえる.

表 2 データ圧縮アルゴリズム別判定率
(ログ単位，属性 URL)

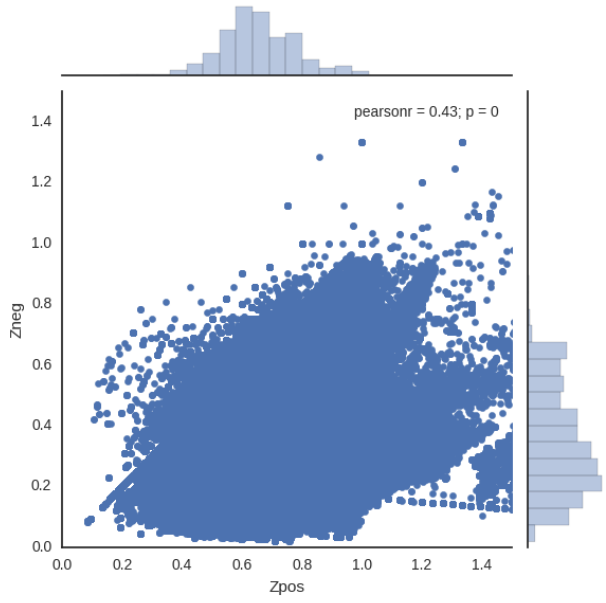
方式	設定等	実行時間	AUC	pAUC	$TPR_{0.5\%}$
LZSS	level 6	16490 秒	0.968	0.0797	60.9 %
LZT	24bit	20 秒	0.973	0.0825	68.1%
bzip2	level 9	68690 秒	0.521	0.0057	0.4 %
LZMA	—	98112 秒	0.975	0.0828	68.3%

5.2 分類器および属性選択の比較

まず，悪性圧縮率と良性圧縮率によって，悪性／良性を分離可能か検討した．検証データの URL 属性から LZT で特徴抽出を行い，横軸にそのデータの悪性圧縮率，縦軸に良性圧縮率をとって検証データをプロットした散布図を図 1 に示す．悪性ログはグラフ左上に，良性ログはグラフ右下にプロットされているのが見られ，線形推論でも十分に分類可能であることがわかった．



a)　悪性ログ



b)　良性ログ

図 1　URL 良性・悪性圧縮率による検証データ散布図

次に FQDN，Path 属性から特徴抽出（特徴ベクトルは 4 次元）を行い，分類器を変えて判定精度を測定した結果を表 3 に示す．L2 正則化 SVM が pAUC 0.0825, $TPR_{0.5\%}$ 65.3% と最も良い値を示し，低誤判定率ではよい分類器であるといえる．

表 4 に属性選択を行い，その判定精度を比較した結果を示す．分類器は L2 正則化 SVM を用いた．URL 全体から特徴抽出する場合は pAUC 0.0720 に対して，FQDN と Path に分けて特徴抽出した場合は pAUC 0.0825 と，2 属性に分けて特徴抽出したほうが良い結果を得られた．

また，図 2 に，判定率 TPR0.5%の経時変化を週ごとに追ったグラフを示す．FQDN,Path 属性での判定は第 1 週では判定率 85%であるが，4 週間ごとに約 10%ずつ判定率が落ちていくことがわかる．また，Path 属性が週次低下していくのに対し，FQDN 属性は 8 週目以降は低下が留まる．

表 3　分類器別判定率
(ホスト単位，属性 FQDN, Path，特徴抽出 LZT 24bit)

分類器	AUC	pAUC	$TPR_{0.5\%}$
Ridge　α =0.1	0.9268	0.0791	59.4%
SVM C=0.025	0.9306	0.0825	65.3%
LDA	0.9291	0.0784	57.3%
NB	0.8166	0.0602	13.5%
AdaBoost	0.9420	0.0785	58.3%
Blacklist*1	—	—	23.2 %
BoW, SVM*2	0.9030	0.0657	32.0 %

*1　対照実験: FPR=0.0%
*2　対照実験: URL を Bag of Words で特徴抽出，C=0.75

表 4　属性選択別判定率
(ホスト単位，特徴抽出 LZT 24bit，分類器 SVM)

属性	AUC	pAUC	$TPR_{0.5\%}$
URL	0.8965	0.0720	41.8%
FQDN	0.8956	0.0631	43.6%
Path	0.7990	0.0562	43.2%
FQDN, Path	0.9306	0.0825	65.3%

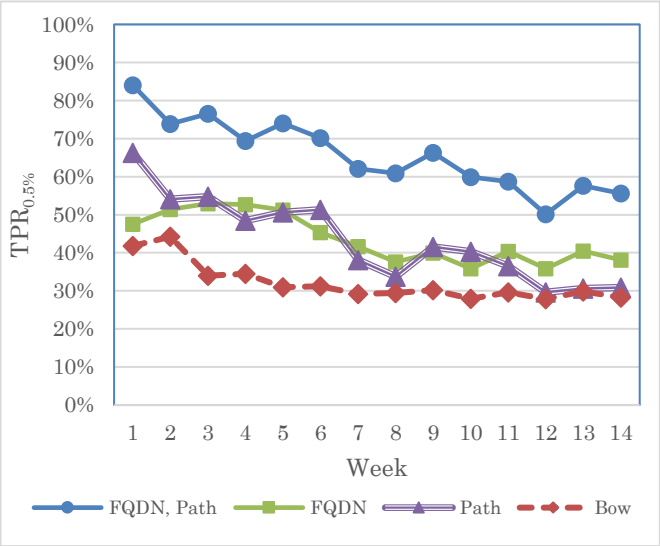


図 2　判定率 $TPR_{0.5\%}$の経時変化
(ホスト単位，特徴抽出 LZT 24bit，分類器 SVM)

6. 考察

なぜデータ圧縮により適切な特徴抽出を行えるのかを以下に考察する．図 3 に URL 属性の悪性圧縮率のヒストグラムを示す．悪性ログのヒストグラムに注目すると，大まかに圧縮率が非常に小さい A，良性ログのピーク値とあまり変わらない圧縮率である B，圧縮率が大きな C の 3 つのピークが存在することがわかる．

A に属する URL として表 1 の 1 行目，2 行目のように，API としてパターン化した URL が存在する．表 1 の 2 行目を実際に LZT で圧縮した状態を表 5 に示す．表の｜はその間のデータ列が圧縮時 1 コードで表されることを示す．未圧縮時に 1,352bit だったデータが，悪性ログとの圧縮を行った場合，220bit にまで圧縮され，かなりのデータ列が，1 コードであらわされるのがわかる．Query 部に注目すると，”&distid=1”のように，ほぼ key 名で 1 コードとなっているものや，”countryid=…”ように value も含めて 1 コードとなっているものがある．すなわち，訓練データに存在しなかった key と value の組み合わせは分割し，存在した組み合わせはセットとなるように，データ圧縮が自動的に認識している．このため，API のようなテンプレート的なデータ列は適切に類似性を認識するように思われる．FQDN での顕著な例は，host1.xxx.com，host2.xxx.com など，ホスト名に連番を含むものが挙げられる．完全一致では異なる FQDN と認識されてしまうが，データ圧縮では，最長データ列一致というデータ圧縮の方針から，|host|2.|xxx.com|のように認識され，類似する(関係エントロピーが少ない)ものとされる．

一方で C に属する URL として，表 1 の 5 行目に見られる，符号化・暗号化を伴う URL が存在する．表 6 にその圧縮状態を示す．これは良性・悪性共に同様に圧縮率が大きくなってしまい，本実験のような線形推論では正しい判定が行えない．しかし，符号化 URL は良性・悪性圧縮率ともに大きいという性質を今後活用することは可能であろう．

LZT のデータ圧縮方針である最長データ列一致が，URL の特徴抽出に適していることは, bzip2 ではまったく判定できないことからも推測される.bzip2 は圧縮時にブロックソートを行うが，その際，データ列をばらばらにして並べ替える．このようにデータ列を認識しない圧縮アルゴリズムが URL での判定を行えないことから，URL の文字列が重要な特徴であることを裏付ける．

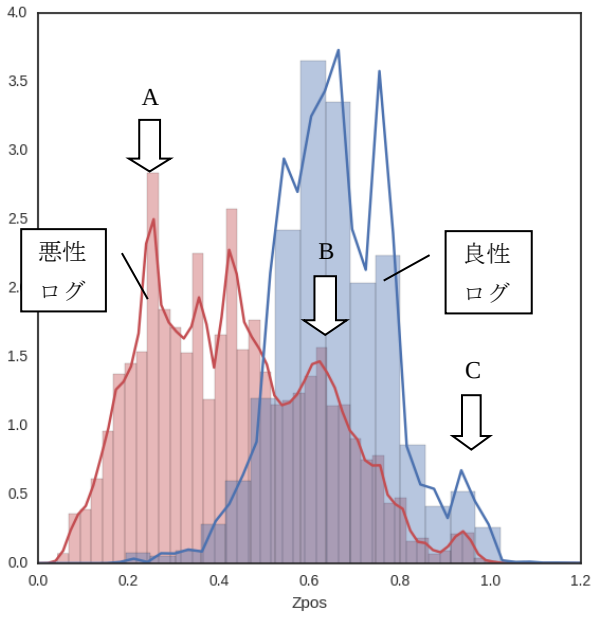


図 3　URL 悪性圧縮率によるヒストグラム

表 5　API 的 URL の圧縮状態

未圧縮: 1352bit: http://serv.the-app-data.info//offers/DynamicOfferScreen?offerid=2&distid=13467&leadp=1694&countryid=142&sysbit=32&dfb=-1&hb=2&isagg=0&version=4.0&external=0&external=0&
良性ログ関連エントロピー: 900bit |http://serv|.the|-ap|p-|data.|info/|/of|fers|/D|yna|mic|Off|erS|cre|en|?o|ffe|rid=2|&dis|tid=1|3467|&le|adp|=16|94&c|ountry|id=14|2&s|ys|bit|=32&|dfb|=-1&|hb=|2&is|agg|=0&ver|sion=|4.0|&ex|ternal|=0&e|xter|nal|=0&|
悪性ログ関連エントロピー: 220bit |http://serv.the-app-data.info//offers/DynamicOfferScreen?offerid=2|&distid=1|3467|&leadp=1694&|countryid=142&sysbit=32&df|b=-1&hb=2|&isagg=0&versio|n=4|.0&e|xternal=0&e|xternal=0&|

* ||の間が圧縮時に 1 コードで表されることを示す．

表 6　符号化 URL の圧縮状態

未圧縮: 4688bit http://d.castplatform.com/api/vp/1?clk=gLg_PHWA9aSyioXkt-F4b3J9cI1ybf-t-x7VxWH5dmAWXwln-z (以下略)
良性ログ関連エントロピー: 4420bit |http://d.|cast|platform|.com/api/|vp|/1|?cl|k=|gLg|_PH|WA9|aS|yio|Xkt|-F4|b3J|9cI|1y|bf-|t-|x7|VxW|H5d|mAW|Xwl|n-|z (以下略)
悪性ログ関連エントロピー: 4980bit A|http://d.|cast|platform|.com/api/v|p/1|?cl|k=|gL|g_|PH|WA|9a|Sy|io|Xk|t-|F4|b3|J9|cI|1yb|f-|t-x|7V|xWH|5d|mA|WX|wl|n-|z

* ||の間が圧縮時に 1 コードで表されることを示す.

7. まとめ

データ圧縮アルゴリズム LZT を用いた特徴抽出により，FQDN 部と Path 部の悪性・良性圧縮率を抽出し，L2 正則化 SVM で学習・推論することにより，ブラックリスト判定や BoW による特徴抽出手法よりも優れた判定精度をしめした. 特に API 的 URL 等のパターン化した URL, FQDN, Path 巻の類似性を認識して特徴抽出を行えるのがデータ圧縮アルゴリズムを用いる利点であると思われる.

一方, 本手法では符号化 URL を含むログをうまく判定することができず，これは今後の課題である.

関係エントロピーの算出は必ずしも悪性・良性両データを必要とするものではなく，どちらか片方のみでも適用可能である．そのため，異常検知手法にも適用できる可能性があり，今後の検討が期待される.

参考文献

[1] Benedetto, et.al., "Language trees and zipping", Phys. Rev. Lett., vol.88, 2002.

[2] Keogh, et.al., "Towards Parameter-Free DataMining", KDD, pp. 206-215 ,2004.

[3] Marton, et.al., "On compression-based text classification", European Conference on Information Retrieval, pp. 300-314 , 2005.

[4] Bratko, et.al., “Spam filtering using statistical data compression”, Journal of Machine Learning Research, vol.7, pp.2673-2698, 2006.

[5] 西田他,「データ圧縮によるツイート話題分類」, 日本データベース学会論文誌, Vol.10, No.1, 2011.

[6] 足達 他、「圧縮類似度による楽譜からの作曲者の判定」、DEIM2013.

[7] T. Nelms, R. Perdisci , and M. Ahamad, “ExecScent : mining for new C&C domains in live networks with adaptive control protocol templates”, 22nd USENIX Conf., pp.589–604, Aug. 2013.

[8] 岡野靖, 熊谷充敏, 谷川真樹, 大嶋嘉人, 愛甲健二, 梅橋一充, 村上純一, 「マルウェア静的判定における誤判定を低減させる誤判定グッドウェアを活用した事例選択手法」, 信学技報, Vol.115, No.224, pp.163-170, 2015.

[9] L. E. Dodd and M. S. Pepe. “Partial AUC estimation and regression.”, Biometrics, 59(3), pp.614–623, 2003.

[10] P. Tischer, “A modified Lempel-Ziv-Welch data compression scheme.”, Aust. Comp. Sci. Commun. 9, 1, pp.262-272, 1987.