データ管理基盤のマルウェア分析への適用

川島 英之†
† 筑波大学 システム情報系
305-8573　茨城県つくば市天王台 1-1-1
kawasima@cs.tsukuba.ac.jp

あらまし 本稿ではD3MおよびPRACTICEデータセットを対象にして，パケット採取システムnegiならびにリレーショナルデータ管理システムPostgreSQLを用いて行った分析結果について述べる．分析の結果，前述のデータセットではTCPハンドシェイクの記録が存在すること，同一destination portへsource portを変えながらアクセスを試みる多数の記録が存在することがわかった．これらの分析はSQLにより実現された．同様の分析をリアルタイムに行うために，データストリーム管理システムを用いる方式について検討を行う．

Applying Data Management Infrastructure to Malware Analysis

Hideyuki　Kawashima†
†Faculty of Information, Systems and Engineering, University of Tsukuba.
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, JAPAN
kawasima@cs.tsukuba.ac.jp

Abstract This describes the result of analysis for D3M and PRACTICE datasets using a packet capturing system NEGI and a relational database management system PostgreSQL. The result of analysis showed that in the above datasets, TCP handshakes exist and enormous accesses to a specific port changing source ports exist. These analytics are done by SQL. This paper also considers to apply a data stream management system to malware analysis to conduct similar analytics in real-time.

1　はじめに

パケットのヘッダ情報はマルウェアの挙動を示す手がかりと考えられる．一方，マルウェア分析に用いられるパケットのデータ量は通常大量である．そこで著者はバッチ処理にはデータベース管理システム(DBMS)の利用が好ましく，リアルタイム処理にはデータストリーム管理システム(DSMS)の利用が好ましいと考えた．そこで本論文はDBMSとDSMSをMWS2013データセットで提供されたパケットヘッダの初歩的分析に利用することを試みる．

本論文ではまずDBMSであるPostgreSQLを利用してパケットの分析を行った結果を述べる．次にDSMSを用いて同様の分析をリアルタイムに行う方式について検討する．本論文の構成は次の通りである．2 節では準備としてパケット採取システムと DBMS について述べる．3 節ではPostgreSQLを用いた分析結果について述べる．4節ではDSMS を用いたリアルタイム分析に関して考察を行う．最後に 5 節では結論を述べる．

2　準備

2.1　パケット取得ツール

パケットヘッダ情報を分析するには，それを採取する必要がある．この目的としては wireshark が有用なツールとして知られているが，我々は negi[1,2]を改変して利用した．Wireshark ではなく negi を利用した理由は，それが改変容易であるからである．我々の研究目的の 1 つはリアルタイムなパケットデータ処理基盤を構築することである．それにはパケット採取システムとデータ処理基盤が密結合していることが好ましい．従って好ましい結合方式は，システムコールをパケット送受信に要する異なるプロセスとしての実装ではなく，ポインタ切り替えによりパケット送受信が実現される，同一プロセス中の別スレッドとしての実装である．そこでコードサイズが小さく安定動作を確認している negi を本研究では用いた．

Negi は pcap 形式のファイル，ならびにネットワークインタフェースからパケットを取り出すことができるツールであ

る. 取得可能なパケットの単位としては IP パケットと TCP パケットが用意されている. また, レイヤ 7 におけるパケットに対してペイロードデータに対する正規表現を用いたフィルタリング機能を提供する. フィルタリング処理を高速化するために, TCP パケットが条件に適合しない場合には, それが構築中であっても破棄する機能を有する. 即ち TCP パケットに対するフィルタリング処理を実行するには, 必ずしも構築を終える必要がない.

ただし本研究においては negi のフィルタリング機能を使用せず, IP パケット取得機能のみを利用した. 取得した情報は次の通りである : パケットの時間情報, source IP, destination IP, source port, destination port, プロトコル番号, シーケンス番号, ACK 番号, syn 情報, ack 情報, fin 情報, urg 情報, psh 情報, rst 情報, IP パケットペイロードサイズ, TCP パケットペイロードサイズ.

2.2 DBMS (PostgreSQL)

DBMS としてリレーショナルデータ管理システム(RDBMS)である PostgreSQL(version 9.3beta2) [3] を本論文では利用した. RDBMS は SQL なる問合せ言語を提供する. ユーザは SQL を利用して問合せを発行し, その結果を受け取る.

データを分析するにあたり, negi が出力したデータを CSV 形式のファイルに変換し, それを COPY コマンドを用いて PostgreSQL に保存した. 用いたデータセットは D3M Datasets 2013 (以下, D3M)と PRACTICE Dataset 2013/CCC Datasets 2013 である (以下, PRACTICE) [4]. D3M に関しては pcap 形式で提供されている URL 情報ならびにハッシュ情報の両方を保存した. テーブル数は 54 になった. PRACTICE については pcap 形式で提供されたデータを保存したテーブル数は 5 となった. すなわち, pcap 形式のファイルを保存した数は合計で 59 となった. テーブルのスキーマは全て等しい. 詳細を 表 1に示す.

表 1 パケットのスキーマ

属性名	型	備考
time	text	時間
srcip	text	Source IP
dstip	text	Destination IP
srcport	integer	Source Port
dstport	integer	Destination Port
protocol	integer	プロトコル番号
seqno	bigint	シーケンス番号
ackno	bigint	ACK 番号
syn	text	SYN フラグ
ack	text	ACK フラグ
fin	text	FIN フラグ
urg	text	URG フラグ
psh	text	PSH フラグ
rst	text	RST フラグ
szpkt	bigint	IP パケットサイズ
szcontent	bigint	TCP パケットサイズ

3 PostgreSQL を用いた分析

3.1 TCP コネクション

3.1.1 問合せ

TCP ハンドシェイクが行われているかを SQL で記述した問合せにより調査した. この問合せは, 我々がストリームリレーションにおいて利用していた問合せ [5] を参考に作成した. 調査に用いた問合せを 図 1 に示す. この問合せはまずパケットを有するリレーションから 2 つ一時的リレーションを生成する. 片方は SYN ラベルを有するリレーション syn であり, もう片方は ACK ラベルを有するリレーション ACK である. また, SYN リレーションと ACK リレーションの結合演算に要するコストを削減するために protocol 番号が 6(TCP を意味)であるタプルのみを選択している. 次に SYN リレーションと ACK リレーションを, srcport と dstport が等しく, srcip と dstip が等しく, そしてシーケンス番号+1 が ACK 番号になっている条件で結合する. これにより TCP ハンドシェイクの開始が行われているパケットの数が得られる.

```
SELECT COUNT(*)
FROM ( SELECT *
        FROM $table
        WHERE syn = 'syn' AND protocol = 6)
        AS syn,
        (SELECT *
        FROM $table
        WHERE ack = 'ack' AND protocol = 6)
        AS ack
WHERE   syn.srcport = ack.dstport
      AND syn.dstport = ack.srcport
      AND syn.srcip = ack.dstip
      AND syn.dstip = ack.srcip
      AND syn.seqno +1 = ack.ackno;
```

図 1 TCP ハンドシェイクの検出

3.1.2 結果

図 1 に示した問合せの結果を表 2に示す. 左端にはリレーション名が示されている. なお, リレーションは最大 60 文字とした為, データセットの名称が一部省略されている. これは PostgreSQL におけるテーブル名の制限による.

検出数はハンドシェイク数を表し, 総数は全パケット数を表す. 表 2をみると, D3M においては最小は 0%, 最大は

35.5%であった．一方，PRACTICE においては最小で 6.7%，最大は 24.9%だった．以上より，D3M は PRACTICE よりも割合の分散が大きいと言える．

表 2 TCP ハンドシェイク数

リレーション名	検出数	総数	割合 (0～1)
D3M_2010_2_pcap_20100308	0	55531	0
D3M_2010_2_pcap_20100309	0	41387	0
D3M_2010_2_pcap_20100311	0	70486	0
D3M_2010_2_pcap_20110208	0	12873	0
D3M_2010_2_pcap_20110214	0	12474	0
D3M_2010_2_pcap_20110216	0	27294	0
D3M_2012_20120321_malware_20120321_33ce026a56df8d5f0a9e9d9eb	0	9	0
D3M_2012_20120321_malware_20120321_4cfab513a206f23ab302e640e	0	32	0
D3M_2012_20120321_malware_20120321_508ea2f818f194c49134f6b2d	0	7	0
D3M_2012_20120321_malware_20120321_52a6dd6ae5c25212ac137d66a	7	72	0.097
D3M_2012_20120321_malware_20120321_545e261b3b00d116a1d69201e	50	192	0.260
D3M_2012_20120321_malware_20120321_9db75a07dea8dfbeeb45785f2	0	11	0
D3M_2012_20120321_malware_20120321_9e43c567ea127b2c18f69730d	0	183	0
D3M_2012_20120321_malware_20120321_fdaf6b9ab4f66940317f86b5c	16	76	0.211
D3M_2012_20120321_url_20120321_url_20120321	1058	12685	0.083
D3M_2012_20120323_malware_20120323_508ea2f818f194c49134f6b2d	0	16	0
D3M_2012_20120323_malware_20120323_7aaed4d93eccd37512af3c6b6	624	2482	0.251
D3M_2012_20120323_malware_20120323_9e43c567ea127b2c18f69730d	0	182	0
D3M_2012_20120323_malware_20120323_fdaf6b9ab4f66940317f86b5c	20	96	0.220
D3M_2012_20120323_url_20120323_url_2012_03_23	464	8844	0.052
D3M_2012_20120325_malware_20120325_493e3f48046d32acc1a3277ea	608	2423	0.251
D3M_2012_20120325_malware_20120325_508ea2f818f194c49134f6b2d	0	9	0
D3M_2012_20120325_malware_20120325_545e261b3b00d116a1d69201e	54	211	0.256
D3M_2012_20120325_malware_20120325_fdaf6b9ab4f66940317f86b5c	16	81	0.198
D3M_2012_20120325_url_20120325_url_20120325	741	8223	0.090
D3M_2012_20120328_malware_20120328_4cfab513a206f23ab302e640e	0	52	0
D3M_2012_20120328_malware_20120328_508ea2f818f194c49134f6b2d	0	11	0
D3M_2012_20120328_malware_20120328_545e261b3b00d116a1d69201e	54	208	0.260
D3M_2012_20120328_malware_20120328_82acd23ea3ca9bad8aa645b50	0	11	0
D3M_2012_20120328_malware_20120328_9e43c567ea127b2c18f69730d	0	183	0
D3M_2012_20120328_malware_20120328_c7d1829504232a8ac5732e3ce	0	10	0
D3M_2012_20120328_malware_20120328_d39569faa627f1aa37ef22ec8	100	8111	0.012
D3M_2012_20120328_malware_20120328_d89d2d390df9b29ab1dacc1e7	540	1522	0.355
D3M_2012_20120328_malware_20120328_fdaf6b9ab4f66940317f86b5c	16	85	0.188
D3M_2012_20120328_url_20120328_url_20120328	901	16682	0.054
D3M_2013_20120802_malware_20120802_545e261b3b00d116a1d69201e	59	223	0.265
D3M_2013_20120802_malware_20120802_9e43c567ea127b2c18f69730d	0	82	0
D3M_2013_20120802_malware_20120802_b78341c653f642f913954473c	6	43	0.140
D3M_2013_20120802_url_20120802_20120802_	263	4713	0.056
D3M_2013_20121002_malware_20121002_194f3fa29470c62bc5a4c6c31	3	616	0.005
D3M_2013_20121002_malware_20121002_2dc24ff6aaa86b9bc4085aacf	6	70	0.086
D3M_2013_20121002_malware_20121002_545e261b3b00d116a1d69201e	52	210	0.248
D3M_2013_20121002_malware_20121002_5f367301a7fda521ee9929b21	48	172	0.279
D3M_2013_20121002_malware_20121002_e2cbd03e982a6c4890a430634	6	73	0.082
D3M_2013_20121002_url_20121002_20121002_	243	4791	0.051
D3M_2013_20130226_malware_20130226_11b78ad63a085e5b5dee12d6b	3	650	0.005
D3M_2013_20130226_malware_20130226_1b5a2bd5d320b3c83f190b4bc	3	671	0.004
D3M_2013_20130226_malware_20130226_4b52ed07c659098768558e2bb	3	665	0.005

D3M_2013_20130226_malware_20130226_54537801e0745bea9ebd1a858	12	56	0.214
D3M_2013_20130226_malware_20130226_5de92a35fee3cbd1fda39e73d	3	628	0.005
D3M_2013_20130226_malware_20130226_9600058e7e7ef522119f799ca	3	650	0.005
D3M_2013_20130226_malware_20130226_b76a94a81aebe736c9023c95f	33	645	0.051
D3M_2013_20130226_malware_20130226_d0b52aa3a71b49725d343b894	0	16	0
D3M_2013_20130226_url_20130226_20130226_	273	9409	0.029
practice_dataset_2013_practice_1	3631	54403	0.067
practice_dataset_2013_practice_2	5393	21680	0.249
practice_dataset_2013_practice_3	192706	1160063	0.166
practice_dataset_2013_practice_4	80560	631453	0.128
practice_dataset_2013_practice_5	5999	49137	0.122

3.2 Srcport と dstport が同一の通信数

Srcport と dstport が同一の通信数を調査した．このような通信を Dstip を変えながら実行するプログラムは特定ポートからの侵入を意図している可能性がある．

3.2.1 問合せ

Srcport と dstport が同一の通信数を調査するために図 2に示されるような問合せを発行した．なお，両図におけるリレーション名ならびに srcip はリレーション毎に適切な値に変更した．

```
SELECT srcport, dstport, dstip, COUNT(*)
FROM practice_dataset_2013_practice_1
WHERE srcip = '10.220.0.36'
GROUP BY srcport, dstport, dstip
ORDER BY COUNT(*) DESC LIMIT 5;
```

図 2 ポート番号が同一である通信の検出

3.2.2 結果

図 2に示される問合せの結果を表 3～8 に示す．表 3～7 は PRACTICE に関する結果である．表 8 には D3M の URL を除く全ての結果の内，最も興味深かった結果を示す．URL に関する結果を除いた理由は，いずれも dstport が 80 である通信が多く，それらはダウンロードに使われていると考えられるからである．

表 3～7 においては同一である srcport と dstport の通信が多数存在したことがわかる．

表 3 同一ポートに関する通信数（PRACTICE-1）

srcport	dstport	dstip	count
68	67	10.220.0.100	324
58360	22543	92.50.8.106	238
61532	80	91.121.95.6	71
59514	22734	45.130.61.242	26
53555	15510	2.134.49.170	18

表 4 同一ポートに関する通信数（PRACTICE-2）

srcport	dstport	dstip	count
68	67	10.220.0.100	324
60432	55003	78.34.188.201	33
59306	55003	78.34.188.201	30
53723	55003	78.34.188.201	30
54299	55003	78.34.188.201	30

表 5 同一ポートに関する通信数（PRACTICE-3）

srcport	dstport	dstip	count
56138	16471	219.80.142.21	1182
56693	16471	219.80.142.21	1182
52717	16471	136.169.13.7	1148
49313	16471	24.15.221.77	1142
58601	16471	98.145.30.7	1057

表 6 同一ポートに関する通信数（PRACTICE-4）

srcport	dstport	dstip	count
61916	16464	189.95.79.14	1149
54793	16464	189.95.79.14	1149
50509	16464	219.55.222.3	1042
63275	16464	219.55.222.3	1042
52628	16464	69.170.93.61	1040

表 7 同一ポートに関する通信数（PRACTICE-5）

srcport	dstport	dstip	count
68	67	10.220.0.100	325
58320	4000	89.149.253.239	45
55567	4000	89.149.253.239	36
58579	4000	89.149.253.239	27
60461	4000	89.149.253.239	27

表 8 同一ポートに関する通信数（D3M_2012_20120328_malware_20120328_d39569faa627f1aa37ef22ec8）

srcport	dstport	dstip	count
1058	80	217.76.142.78	1401
1060	80	81.31.32.70	1376
1055	80	213.153.32.161	767
1054	53	10.0.0.1	67
1040	8082	186.5.23.154	7

3.2.3 考察

表 3～7 より，srcport と dstport が同一である通信が多数存在することがわかった．この理由として筆者は次の可能性があると考える．1：他ホストへの侵入を狙った可能性．2：自プロセスの存在を伝える信号を送っていた可能性．ただしこれらの推測を支える根拠は得られていない．

3.3 Dstport が同一で srcport が異なる通信

3.3.1 問合せ

3.2.3 節では侵入を狙った痕跡である可能性が PRACTICE データセットに存在すると述べた. そこで srcport を変えながら同じ dstport へアクセスするようなトラフィックの存在について調査した. 問合せには図 2に示されるような問合せをもちいた. ただし LIMIT を削除して全ての結果を見た.

3.3.2 結果

PRACTICE-1 においては srcport を変えながら特定 dstport にアクセスする現象が観察されなかった.

PRACTICE-2 においては dstport 55003 に対して, srcport を変えながらアクセスする現象が 1535 回観測された. なお, srcport を変えながらアクセスする現象は全体で 1868 回であった. 55003 以外では 53 が支配的であり, その現象は 330 回だった.

PRACTICE-3 においては dstport 16471 に対して, srcport を変えながらアクセスする現象が 16383 回観測された. なお, srcport を変えながらアクセスする現象は全体で 16484 回であった.

PRACTICE-4 においては dstport 16464 に対して, srcport を変えながらアクセスする現象が 16382 回観測された. なお, srcport を変えながらアクセスする現象は全体で 16492 回であった.

PRACTICE-5 においては dstport 4000 に対して, srcport を変えながらアクセスする現象が 3685 回観測された. なお, srcport を変えながらアクセスする現象は全体で 5859 回であった. 他は dstport が 80 の現象が 1821 回, そして dstport が 53 の現象が 350 回だった.

3.3.3 考察

Srcport を変えながら特定の dstport にアクセスする現象は PRACTICE-1 では観察されなかったが, PRACTICE-2～5 では観察された. 提供された資料によれば, PRACTICE-1～5 はいずれもトロイの木馬であるとのことだが, 挙動が必ずしも同じではないことが観測結果からわかる. 従って, トロイの木馬には同一 dstport へのアクセスを行う種と行わない種が存在すると考えられる.

また, 今回の観測結果からは同一 dstport へのアクセスを試みる傾向は 80%であった. 従って今回提供されたトロイの木馬には存在したと言える. ただし今回提供された記録は高々5 種の検体の記録である. 従ってこの傾向がトロイの木馬全般について言及できるか否かは不明である.

3.4 パケットの流量(外から内)

マルウェアが外部と通信し, 外部からの攻撃を手引きする場合には, 当該ポートへのデータ送信量や送信回数が増大すると考えられる. そこでパケットの流量を分析する.

3.4.1 問合せ

```
SELECT dstport, SUM(szcontent), AVG(szcontent),
COUNT(*)
FROM practice_dataset_2013_practice_1
WHERE dstip = '10.220.0.36'
GROUP BY dstport
ORDER BY SUM DESC LIMIT 5;
```

図 3 パケット流量(外から内)を求める問合せ

3.4.2 結果

分析結果を表 9～16 に示す. 表 9～13 は PRACTICE に関する結果であり, 表 14～16 は D3M に関する結果である. D3M は顕著な結果を示す 3 例のみを示している.

表 9～16 より, PRACTICE においては特定のポートに対して大量のデータが送信されていることがわかる. PRACTICE-1 では 61532(不明)と 68(bootpc), PRACTCE-2 では 68(bootpc)と 53050(不明), PRACTICE-5 では 68(bootpc)である.

表 13～16 より, D3M においては特定のポートに対して大量のデータが送信されていることがわかる. それらは 1034(activesync), 1033(netinfo-local), 1058(nim), 1060(polestar), 1055(ansyslmd)である.

表 9 パケット流量(外から内) (PRACTICE-1)

dstport	sum	avg	count
61532	125996	1259	100
68	103620	314	330
59514	22532	901	25
50705	12622	841	15
60472	10096	631	16

表 10 パケット流量(外から内)(PRACTICE-2)

dstport	sum	avg	count
68	103620	314	330
53050	1516	151	10
63441	1416	71	20
60341	1062	71	15
60472	1062	62	17

表 11 パケット流量(外から内)(PRACTICE-3)

dstport	sum	avg	count
62222	106182	564	188
50925	105820	534	198
53796	95232	566	168
60031	84972	1075	79
56175	84434	1042	81

表 12 パケット流量(外から内)(PRACTICE-4)

dstport	sum	avg	count
64413	57274	970	59
54313	53310	987	54
60403	52135	930	56
60844	50918	893	57
58837	50856	978	52

表 13 パケット流量(外から内)(PRACTICE-5)

dstport	sum	avg	count
68	103934	314	331
53050	1516	151	10
56361	652	163	4
59994	520	52	10
123	496	62	8

表 14 パケット流量(外から内)
(D3M_2012_20120323_malware_20120323_7aaed4d93eccd37512af3c6b6)

dstport	sum	avg	count
1034	14241	64	221
1190	975	163	6
1126	325	54	6
1095	325	54	6
1106	325	54	6

表 15 パケット流量(外から内)
(D3M_2012_20120325_malware_20120325_493e3f48046d32acc1a3277ea)

dstport	sum	avg	count
1033	13965	64	217
1126	325	54	6
1095	325	54	6
1106	325	54	6
1112	325	54	6

表 16 パケット流量(外から内)
(D3M_2012_20120328_malware_20120328_d39569faaa627f1aa37ef22ec8)

dstport	sum	avg	count
1058	822414	509	1654
1060	781472	510	1530
1055	489296	509	960
1054	4684	73	64
1035	325	54	6

3.4.3 考察

Dstport 68 (bootpc), 1034 (activesync), 1033 (netinfo-local), 1060(polestar), 1055(ansyslmd)はいずれも DoS 等の攻撃に使われる可能性があると考えられる. 解析結果より, これらのポートへのデータ量ならびにアクセス回数が多いことがわかった. この理由としてマルウェアが外部からの攻撃を手引きしていた可能性がある.

3.5 パケットの流量(内から外)

3.5.1 問合せ

```
SELECT dstport, SUM(szcontent), AVG(szcontent),
COUNT(*)
FROM practice_dataset_2013_practice_1
WHERE srcip = '10.220.0.36'
GROUP BY dstport
ORDER BY SUM DESC LIMIT 5;
```

図 4 パケット流量(内から外)を求める問合せ

パケット流量(内から外)を求める問合せを図 4 に示す. 図 3 との違いは WHERE 句の dstip と srcip が変わっている点のみである.

3.5.2 結果

分析結果を表 17～24 に示す. 表 17～21 は PRACTICE に関する結果であり, 表 22～24 は D3M に関する結果である. D3M は顕著な結果を示す 3 例のみを示している.

PRACTICE に関する結果(表 17～21)より, 次のポートへの流量が多いことがわかる: 53(domain), 55003(不明), 16471(不明), 16464(不明), 80(http).

D3Mに関する結果(表21～24)より, 次のポートへの流量が多いことがわかる: 8080 (webcache), 137 (netbios-ns), 8082(us-cli).

3.5.3 考察

表 23 では全く同一の結果が 3 つのリレーションから得られた. リレーションの名前から, これらは同一のマルウェアの異なる日における記録だと考えられる. ある種のマルウェアについては 24 時間周期で同様な挙動を繰り返すのかもしれない.

表 24 では sum は小さいが count が大きい例が見られた. 紙面の都合から省略するが, このような例は他にも見られた. これより, 異常現象を見つけるためには count だけ, あるいは sum だけのように 1 つの分析結果のみを見ていては不足であると考えられる. 様々な分析手法を 1 つのパケットストリームに対して同時に適用することが望ましいと考えられる.

表 17 パケット流量(内から外) (PRACTICE-1)

dstport	sum	avg	count
53	138413	46	3016
25549	104778	177	591
67	102364	314	326
18746	61375	164	373
26614	55162	81	680

表 178 パケット流量(内から外)(PRACTICE-2)

dstport	sum	avg	count
55003	114926	9.5	12067
67	102364	314	326
53	15424	45	342
123	496	62	8
80	378	18	21

表 19 パケット流量(内から外)(PRACTICE-3)

dstport	sum	avg	count
16471	5789202	8.43	686648
67	29830	314	95
53	4288	45	95
80	336	17	20
123	248	62	4

表 20 パケット流量(内から外)(PRACTICE-4)

dstport	sum	avg	count
16464	5319348	14	375582
67	29830	314	95
53	4288	45	95
80	1846	23	80
123	248	62	4

表 21 パケット流量(内から外)(PRACTICE-5)

dstport	sum	avg	count
80	222693	37	6202
67	102678	314	327
53	16580	45	362
123	496	62	8
99	0	0	18

表 22 パケット流量(内から外)
(D3M_2012_20120323_malware_20120323_7aaed4d93eccd37512af3c6b6)

dstport	sum	avg	count
8080	50076	46	1089
53	11872	53	224
138	663	221	3

表 23 パケット流量(内から外)
(D3M_2012_20120321_malware_20120321_9e43c567ea127b2c18f69730d
D3M_2012_20120323_malware_20120323_9e43c567ea127b2c18f69730d
D3M_2012_20120328_malware_20120328_9e43c567ea127b2c18f69730d)

dstport	sum	avg	count
137	10368	64	162
138	663	221	3
53	228	46	5

表 24 パケット流量(内から外)
(D3M_2012_20120328_malware_20120328_d39569faa627f1aa37ef22ec8)

dstport	sum	avg	count
8080	4917	64	77
8082	4895	64	77
53	4759	71	67
138	1308	218	6
80	731	0.2	3550

3.6 異常検知

上記では RDBMS が提供する分析手段(具体的にはリレーショナル演算子)により分析を行った. RDBMS には存在しない分析手法も多数存在する. そこで異常検知手法として知られている LOF (local outlier factor)を本節では適用してみる.

3.6.1 LOF (Local Outlier Factor) [6]

LOF は密度に基づく異常検知手法である. 距離に基づく異常検知手法として DBO (Distance Based Outlier)が存在するが, DBO はパラメータを 2 つ取るのに対して LOF はパラメータを 1 つしか取らないので使いやすい. LOF は非異常点に関する LOF のスコアはおよそ 1 になり, 異常点については 1 よりも大きくなる. LOF は DBO よりも高い検出性能を示すことが知られている. 本研究では文献[6]を参考にして C++により LOF を実装した.

実験ではパラメータ minpts を 10 にして PRACTICE-1 に LOF を適用した. Srcport と dstport のタプルから成るデータを入力データとした. 即ち二次元のデータを LOF の入力とした.

3.6.2 結果

実験の結果, スコアが 30 を超えるタプルが 44 件出力された. なお, PRACTICE-1 は 54403 件である. 上位 10 件の結果を表 25 に示す.

表 25 LOF による PRACTICE-1 の分析結果

srcport	dstport	LOF スコア
54472	25197	169
52061	29700	134
54591	25412	102
54591	25412	102
49577	2491	99
62235	17878	94
63396	29300	94
51168	16882	74
60708	28777	73
62154	29686	72

3.6.3 考察

表 25 に示される実験結果を観察しても, 何らかの知見を得ることは難しい. 理由の 1 つは, これらの通信が全てマルウェアによりなされているからだと考えられる. 殆どのデータが正常であり, マルウェアに生成されたデータが少しだけある場合に異常検出手法は有効かもしれない. また, 入力データとして srcport, dstport の 2 次元データセットを用意したが, もっと次元の高いデータを準備することが望ましい可能性はある.

4　DSMS の利用に関する検討

3 節では DBMS を用いたパケット分析について述べた．DBMS を用いる分析では，データを一旦 DBMS に格納し，それから分析を行わざるを得ない．それゆえ分析時間が長くなる傾向がある．一方 DSMS ならば，メモリに到着したデータに対してウィンドウを適用し，到着データに対して即座に微視的分析を実行する．マルウェア分析は侵入検知の観点からはリアルタイム性が高いことが望ましい．

商用 DSMS には学術目的ならば無償利用できるシステムも存在する[8]．そのようなシステムの利用が望ましい．

我々はリレーショナル演算子に加えて LOF，DBO，頻出アイテム集合マイニング，変化点検出（change point detection），ベイジアンネットワーク，複合イベント検出（CEP）等の様々なマイニング演算子を有する DSMS [7,9]を開発している．我々のシステムを図 5 （[7]より抜粋）に示す．

In-DSMS 方式が優れる点は来歴管理である．システムが異常であると通知してきた場合に，その理由として来歴，すなわち，どのような入力データに対してどのような処理を施したかに関する記録，を閲覧することは管理者がシステムの通知を信ずるか否かに関して重要な判断材料となり得る．来歴管理機能が DSMS 外部にある場合には DSMS と来歴管理機構が頻繁に通信せざるを得ない．また，来歴の粒度が大きくならざるをえないため，来歴に関する情報量が削減する可能性がある．

DSMS には[8]のようなリレーショナル演算子を中心としてリアルタイム分析を行うシステムも存在するが，更に高度な分析を手軽，高性能，高信頼に実行するには In-DSMS 方式を採用したシステムの利用が好ましいと考える．ただし現状では安定動作するシステムは限られていると考える．

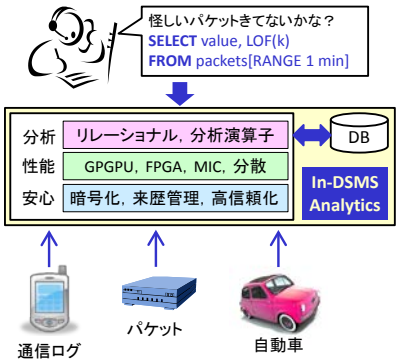

図 5：In-DSMS 方式に基づく DSMS（[7]より抜粋）

5　結論

本論文では D3M と PRACTICE を分析するために RDBMS である PostgreSQL を利用した．分析の結果，SQL を用いるだけで，ある程度の知見を得られることが示された．知見の例としては，同一 srcport と dstport の通信が多数存在すること，特定 dstport への srcport を変えながらの多数のアクセスが存在すること，様々な集約関数は有益であるが，複数を同時に使うべきであること，攻撃と判断されうる記録が存在すること，等である．また，異常検出手法である LOF を，srcport と dstport を属性とする二次元データに対して適用した．その結果，異常性を示唆するタプルが検出されたが検証はできなかった．

今後の課題は，更に興味深い知見を得るために，LOF 以外のマイニング演算子を適用することと，リアルタイム分析を実現するために In-DSMS 方式に基づく DSMS を利用することである．また，今回得られた知見はシステム内部に存在するマルウェアの分析により得られた．今後はシステムが外部から攻撃された場合について分析したい．

謝辞

貴重なデータセットを利用させてくださった MWS ‘13 実行委員会各位ならびにデータセット提供者に感謝する．

参考文献

[1] “Negi”, https://github.com/westlab/negi

[2] 石田 慎一，原島 真吾，鯉渕 道紘，川島 英之，西 宏章，“トラヒックからアプリケーションレイヤ情報の検索・抽出を可能とするソフトウェアの実装と評価”，ソフトウェア科学会論文誌，Vol. 29, No. 3, pp.59-73, 2012-11-01.

[3] PostgreSQL, http://www.postgresql.org/

[4] マルウェア対策研究人材育成ワークショップ 2013 (MWS2013), http://www.iwsec.org/mws/2013/

[5] Yasin Oge, Takefumi Miyoshi, Hideyuki Kawashima, and Tsutomu Yoshinaga. 2013. A fast handshake join implementation on FPGA with adaptive merging network. In Proceedings of the 25th International Conference on Scientific and Statistical Database Management (SSDBM).

[6] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. “LOF: identifying density-based local outliers.” SIGMOD '00.

[7] 川島英之，“In-DSMS 分析システムへ向けて，”情報処理学会研究報告．UBI，[ユビキタスコンピューティングシステム] 2013-UBI-38(3), 1-6, 2013-05-09.

[8]「uCosminexus Stream Data Platform」貸し出しによるストリームデータ処理技術の研究支援 http://www.hitachi.co.jp/Prod/comp/soft1/cosminexus/sdp/academic.html

[9] Oke Masahiro, Hideyuki Kawashima. 2013. “A Multiple Query Optimization Scheme for Change Point Detection.” BIRTE’13 in conjunction with VLDB.