VISUALIZING NETWORK STATUS

KENICHI YOSHIDA[1], YUSUKE SHOMURA[2], YOSHINORI WATANABE[3]

[1]Graduate School of Business Science, University of Tsukuba, Bunkyo, Tokyo 112-0012, Japan.
[2]Central Research Laboratory, Hitachi Ltd., Kokubunji, Tokyo 185-8601, Japan.
[3]ALAXALA Networks Corporation, Kawasaki, Kanagawa 212-0058 Japan
E-MAIL: yoshida@gssm.otsuka.tsukuba.ac.jp

Abstract:

The recent evolution of various communication devices makes both network monitoring and configuration difficult. Although these network management tasks require deep understanding of network status, various new network devices make it difficult by increasing the complexity of whole network system. This paper proposes a visualization technique of network status that uses a frequent itemset mining algorithm to find important phenomena in the network. We also show that a simple interface with the proposed technique can visualize not only the ordinal network status but also the various security incidents. The understanding of the ordinal network status and the rapid finding of security incident helps the naive users in monitoring and configuring network equipments. Finding of various security incidents from real Internet traffic data are described as the typical usage of the proposed visualization technique.

Keywords:

Network monitoring; Data mining

1. Introduction

The recent evolution of various communication devices makes both network monitoring and configuration difficult. Although these network management tasks require deep understanding of network status, various new network devices make it difficult by increasing the complexity of whole network system. This paper proposes a visualization technique of network status that uses a frequent itemset mining algorithm [1] to find important phenomena in the network.

The frequent itemset mining algorithm used in this study finds frequently occurring phenomena as the ordinal network status. It also analyses the variety in the found itemsets. More precisely, after mining frequent itemsets, the non-frequent parts of original data of found itemsets and the numbers of their varieties are calculated and analysed. This additional analysis can visualize not only the ordinal network status but also the various security incidents. The understanding of the ordinal network status and the rapid finding of security incident helps the naive users in monitoring and configuring network equipments.

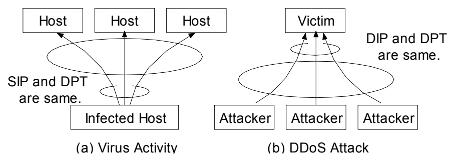

Figure 1. Behavior of Problematic Flows

Figure 1 shows the basic idea. Problematic flows such as P2P flows and internet worms tend to generate many TCP/IP packets, which share the same combination of TCP/IP header information. Thus, frequent itemset mining seems to be a promising approach to find them. More precisely, Internet viruses search for vulnerable hosts by exhaustively accessing various IP addresses (Figure 1 (a)). Such virus behavior tends to create many packets that share the same combination of source IP address (SIP) and destination port number (DPT). Here the SIP is the address of hosts that are already affected by an Internet virus and are trying to find the next victims. The DPT is the port number of the target services, which the virus is trying to use for its intrusion. Note that such flows have various destination IP address (DIP). Since such varieties in the DIP are not observed in the normal combination of SIP and DPT, the proposed analysis can identify computers which are affected by computer viruses.

DDoS attacks follow a similar behavior as discussed above. They attempt to make a computer resource unavailable by a flood of packets with a forged sender address (Figure 1 (b)). Such attacks tend to send many packets that share the same combination of DIP and DPT. Again, such flows have varieties of SIP. These behavior of problematic flows and the analysis of them are the typical examples which show the fact that the analysis of the non-frequent parts for the found itemsets helps the understanding of the network status.

Thus the frequent itemset mining algorithm [1] is a promising approach to analyze high frequency TCP/IP header information being caused by infected computers. Because monitoring internet data is an on-line activity, stream mining techniques [2] seem to be more appropriate. The following research will continue this line of thinking. We also propose a

1-4244-0973-X/07/$25.00 ©2007 IEEE

simple and one-pass stream mining algorithm which can simultaneously count the number of varieties in the found itemsets.

The organization of this paper is as follows. Section 2 briefly reviews related work. Section 3 explains the proposed method, and Section 4 reports on the experimental results. In Section 4, finding of various security incidents from real Internet traffic data is described as the typical usage of the proposed visualization technique. Section 5 discusses related issues, and Section 6 summarizes the findings of the research.

2. Related Work

The monitoring of Internet traffic is an extensively studied area. For example, IETF's IPPM working group proposes a framework of IP performance metrics [3]. Their work is important in providing a baseline to compare measured results by standardizing the attributes to be measured. MAWI [4] provides an archive of actual Internet traffic data. Analysis of measured data has also been studied. Some studies, e.g., [5] and [6], try to use data-mining techniques [1, 2] to automate the analysis.

[7] reports the importance of the number of distinct elements in TCP/IP headers in network analysis. For example, P2P software generates its service port numbers randomly, and its packets include various DPTs while retaining the same SIP. Thus, P2P packets can be identified by finding groups of packets that share the same SIP but have a variety of DPTs. However, this method require a significant amount of memory to analyze the vast amount of data in Internet backbones.

Several studies have reported methods that collect the number of distinct flows efficiently, e.g., [8] and [9]. However, these methods lack the ability to find arbitrary itemsets. These methods can count only the number of specific flows.

Among conventional studies, the techniques of CPM [5] and Space-Saving [10] utilize restricted memory resources to capably deal with a vast amount data. The method proposed in [5] has the ability to find candidates of problematic flows under limited memory constraints. Because of its simplicity, [5] was chosen as the base implementation of the proposed method. However, the ability to analyze the varieties in the found itemsets was added.

3. Algorithm

To analyze the found frequent itemsets and count the number of their varieties, a modified CPM [5], which uses a fixed size cache memory to find frequent itemsets was used. For example, when the modified CPM measures a frequent combination of SIP and DPT, it also counts the variety of other elements in TCP/IP headers, e.g., DIP and source port number (SPT). The information about the number of distinct DIPs and SPTs enables the classification of host behavior.

The algorithm of modified CPM is shown in Figure 2. The underlined part shows the modification. The remainder is the original CPM algorithm. The original CPM algorithm is simple. It simply counts the frequency of itemsets (i.e., possible combinations of items) in the transactions. In the proposed applications, each transaction is made from the TCP/IP header of every single packet. A fixed size cache was used to store the itemsets and their frequencies. Although the basic algorithm is extremely simple, this algorithm works well with a specific memory management strategy.

```
Algorithm CPM
Variable
    Cache []: Fixed Size Table
begin
    Create empty cache;
    while (input Transaction)
        for each item in Transaction
            Itemsets(item, rest of items in Transaction);
end

Function Itemsets(items, rests)
Variable
    items[]: items in the current itemsets
    rests[]: other items in the transaction
begin
    i = index of items in cache;
 if (i is new index)
  increment cache_diff[index of riginal items] by 1;
    increment cache_cnt[i] by 1;
    for each item in rests
        Itemsets(items + item, rest of items in rests);
    if (cache_cun[i] ≥ thresh_hold)
        rport statistics;
        cache_cnt[i] = 0;
        cache_diff[i] = 0;
end
```

Figure 1. Algorithm

```
Function Hash2
Input
    Item: Data to be stored in Cache
Variable
    Hash[]: Table of Hash Values
    Idx[]: Table of Cache Index
begin
    Calculate "n" hash values from Item and store them into
    Hash[]
    Idx[] = Hash[] % Cache Size
    if (one of entry refereed by Idx[] stores Item)
    then return Idx that refers the entry
    else Select Idx that refers least frequent entry
        cache_cnt[Idx] = 0
        return Idx
end
```

Figure 2. Pseudo code of Memory Management

The Hash2 memory management strategy is applied (Figure 3), which is also proposed in [5]. Hash2 first calculates "n" hash values of a given item. Next it generates "n" indexes from "n" hash values. If the item to be stored is already in the cache, one of the indexes refers to the entry for the item. If the item is a new item, then Hash2 selects the index that refers to the least frequent entry out of "n" entries referred to by "n" indexes. Then, the old item stored in the cache at that index will be replaced by the new item.

To analyze the varieties in found itemsets, steps (single-underlined in Figure 2) are added. When the method selects an entry to be stored, it checks whether the combination of items is new. If the combination is new, the method incrementally increases the number of varieties. For example, if a combination of a SIP (as a variable items in Figure 2) and a DPT (as an element of variable rests) is new, the method incrementally increases the counter which stores the number of distinct DPTs.

Table 1. Structure of Cache Table

ID	cache _cnt	1st Item(e. g. SIP)		2nd Item(e. g. SPT)		n-th Item
		cache_diff	value	cache_diff	value	…
1	16	-	192.168.1.5	-	80	…
2	152	-	10.100.0.19	8	-	…
3	2	2	-	-	443	…
4	40	-	172.16.1.46	-	110	…

The analysis of internet traffic data examined in this paper differs from standard frequent itemset mining problems in that each item in the transaction has a specific meaning, such as SIP and SPT. A cache table with fixed entries is used to store the information (See Table 1). It stores the counters for the varieties (cache_diff in Table 1) together with the frequency of itemsets (cache_cnt in Table 1). Actually, the cache table used in the experiments described in the next section has columns for five items, i.e., SIP, SPT, DIP, DPT, and protocol number.

Each entry, i.e. each row in Table 1, stores the value of the item or the number of its varieties. More precisely, each column of each row stores the value of the specific item ("value" column in Table 1) or the number of its varieties ("cache_diff" column in Table 1) in exclusive way. When it stores the "value", the "cache_diff" is empty. And when it stores the "cache_diff", the "value" is empty. When the cache table for 5 items is used and the entry stores the information of the itemset with 2 items, 2 corresponding columns store the values of items and other columns store the number of varieties of the corresponding items. For example, the first data in the table (ID=1) shows:

- There are 16 packets whose SIP is "192.168.1.5" and SPT is "80".

The second data in the table (ID=2) shows:

- There are 152 packets whose SIP is "10.100.0.19".
- The number of varieties in SPT of these packets is "8".

A normalized operation has also implemented. The second modified step (doubly-underlined in Figure 2) implements the normalized operation. The modified step reports the statistics of frequent items when their frequency is greater than the threshold. After the statistics are reported, the entry is set to zero. Through this operation, statistics on the same number of related transactions are collected and normalized information about their varieties are obtained.

4. Experimental Results

By using the proposed method, the number of distinct IP addresses, port numbers, and their combinations from actual Internet traffic logs were found. As a result, problematic traffic such as Internet worms, port scans, and P2P traffic were also visualized. The following section, describes a portion of the results. The traffic log was recorded at a monitoring point of a line crossing the pacific ocean and was provided by the MAWI working group [4]. The characteristics of the traffic flow are summarized in Table 2.

Table 2. Summary of Traffic

	Trans-Paific line
Number of Packets	16,031,975
Number of Flows	1,228,425
Percentage of SYN packets	5.27%
Average Packets Length	159.56

4.1. Overview

Based on the design described in the previous section, a prototype system has been implemented using a personal computer with a Xeon 2.0GHz CPU and 2Gbytes of RAM memory. The system can visualize various information about packet flows appearing more than 1,000 times. This information allows for a variety of analyses, as can be seen by the diversity of data shown in Figure 4. Each data point in the figure indicates the number of distinct SPTs, DPTs, and DIPs for each SIP, and each SIP corresponds to a single client computer connected to the Internet. Thus Figure 4 shows the distribution of client computer characteristics.

As shown in the Figure 4, we can visualize the network status by the proposed method. The shape of the flow distribution seems to vary from network to network. The applications used in the network affects the shape. Thus the shape represents ordinal network status. The problematic flows also affects the shape. The next subsections discuss how we can use these information to analyze problematic flows in details.

The prototype system took 110.9 seconds to process all the traffic data. That is 141 k packets per second, i.e., 1.60 M packets per 110 seconds. If packets with the SYN flag only were analysed, the performance is suited to monitor 3.42 Gbps (141 kpps * 159.56 byte * 8 bits / 0.0527) line traffic. The analysis of packets with the SYN flag reveals valuable information (see next subsection). The average use f backbone bandwidth is far less than 100%, therefore the performance of the proposed method could potentially handle real internet traffic of up to

10-Gbps.

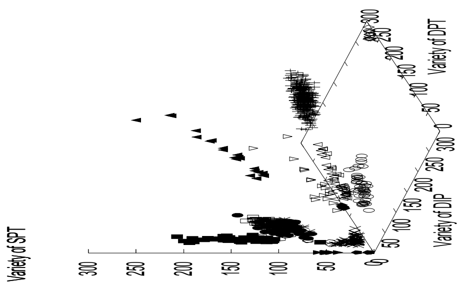

Figure 4. Number of distinct SPTs, DPTs, and DIPs

Table 3. TCP packets with SYN flags classification

Variety of DPTs		# of SIP	Collected Flows	Flows(%)
1, 2	Scan	2	58,870	11.12
	Worm	8	3,958	0.74
	Normal	5	7,033	1.31
3-10	Normal	6	36,530	6.78
	Unknown	5	7,821	1.45
11-100	Normal	1	7,370	1.37
	Unknown	2	9,933	1.85
100-	P2P	1	3,971	0.74
Total		44	155,580	28.89

In the experiments described in the following subsections, a cache table with 12 M entries was used. Each entry consumes 50 bytes of memory, so the cache table contains 600 M byte. With this 600 M byte cache table, all 254 addresses scanned by Internet worms were found (see next subsection). That corresponds to a substantial number of scanned C class addresses. This result indicates that 1) the proposed method found an accurate number of varieties in frequently found flows, and 2) 600 M bytes were sufficient to analyze actual Internet traffic. See [5] for a detailed discussion on the memory consumption characteristics of the proposed method.

4.2. Analysis of TCP Packets with SYN Flag

By reviewing TCP packets with SYN flags, 28.9% of the flows were analyzed. The results are summarized in Table 3. The table shows the number of the distinct DPTs for packets that share the same SIP. Since the number of the distinct DPTs for a specific SIP exhibits the behavior of the client computer with that IP address, the table summarizes the types of client computers in the network. For example, the first row in Table 3 indicates that two hosts (2 SIP, third column) generate 59,870 flows to a specific 1 (or 2) DPTs (first column). The 8th row indicates that a single host generates 3,971 flows to various (over 100) DPTs

Since the number of the distinct DPTs indicates the number of services the SIP node tries to get, the second column indicates the service. The actual service of each flow is confirmed by the following characteristics. Note that finding these services based on the results shown in Table 3 is simple.

Scan: It was found that the number of the distinct DIPs for these flows is more than 500. These flows only contain packets with the SYN flag and do not contain data packets. These flows indicate the typical behavior of crackers or viruses trying to find victims.

Worm: The number of the distinct DIPs for these flows was 254. These flows only contain packets with the SYN flag and do not contain data packets. These characteristics indicate the typical behavior of Internet worms that scan a C class network segment.

P2P: The number of the distinct DPTs is more than 100. This behavior indicates that the hosts generate service port numbers randomly. They also use the combination of the same TCP port and UDP port. This is the typical behavior of the BitTorrent client. This was also confirmed by observing the header portion of payloads.

Normal: These are flows that use well-known services, such as SMTP, HTTP, and SSH.

Unknown: Other flows.

The total number of flows found by this analysis of TCP packets with SYN flags is small. There are 44 flows, of which there is a high probability that problematic flows can be found. These results indicate the effectiveness of the proposed method in finding problems in network traffic.

4.3. Analysis of All TCP Packets

By analyzing all the TCP packets, another problematic flow was found. Typical results of the TCP packets analysis are shown in Figure 5. The number of packets that have the combination of the same SIP and SPT are shown. In this figure, the X-axis corresponds to the number of distinct DIPs, and the Y-axis corresponds to the number of distinct DPTs. For example, (d) in the figure indicates that:

- The number of the distinct DPTs for packets from a single port of a single host is about 600.
- The number of the distinct DIPs for the same packets is also about 600.

The figure shows the statistics collected on a combination of the same SIP and SPTs, which mainly demonstrates server output behavior. These statistics can be classified into three groups according to the number of distinct DIPs.

500≤Variety: Four combinations of the same SIP and SPT are found (see (a),(b),(c),(d) in Figure 5). These four hosts try to find victims of the attack. Host (a) only sends packets with RST and ACK flags, and it always uses SPT 7000. Host (b) only sends packets with SYN and ACK flags, and it also uses SPT 7000. Host (c) and (d) only send packets with SYN and ACK flags, and they use SPT 80.

150≤Variety < 500: Two combinations of the same SIP and SPT are found in this category (see (e) and (f)). They also use the combination of the same TCP port and UDP port.

These two hosts seem to be the hub hosts of the overlay network. They try to maintain sessions with a lot of hosts to maintain an overlay network.

Variety < 150: Other flows in which any conclusion couldn't be determined by this particular analysis.

Clearly, a group of packets whose DIP variety is greater than 150 tends to have problems and is worth monitoring.

4.4. Analysis of UDP Packets

By analyzing UDP packets in similar ways, problematic hosts were found that scan victim hosts using UDP and hosts that form a P2P network. Figure 6 shows the typical results. Statistics of the packets that shared the same SIP and SPT are shown in these figures. The X-axis indicates the number of distinct DIPs, and the Y-axis indicates the number of distinct DPTs. From these results, 7 clusters of DPT and DIP combinations were found (see Figure 7).

Based on the results shown in Figure 6, a manual analysis was performed. Table 4 shows our findings:

Area (1): Flows in this area are considered to be real-time traffic such as voice and video. Manual analysis did not find any strange behavior in the flows in this area.

Area (2): Most of the flows in this area are DNS client traffic. They are packets going to DPT 53, i.e., DNS servers. Since DNS clients connect to multiple DNS servers with the same SPT, a variety of DIPs are seen in a single combination of the same SIP, SPT, and DPT.

Area (3): Scans and spam were detected in this area. Scans were caused by SQL slammer which uses port number 1434. Spam used the Windows Messenger service which uses port numbers 1026 and 1027.

Area (4): DNS server traffic in this area was detected. The traffic is output from a DNS server. Broken packets and P2P traffic were also found. Broken packets are located near the Y-axis. P2P traffic is located at y < x, which is far from the points of the DNS server.

Area (6): DNS server traffic in this area was detected. The traffic consisted of packets from SPT 53.

Area (7): All the flows in this area are those of P2P traffic. They use the combination of the same TCP port and UDP port, which is typical behavior of P2P software.

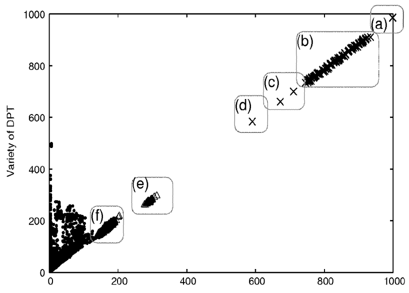

Figure 5. All TCP packets

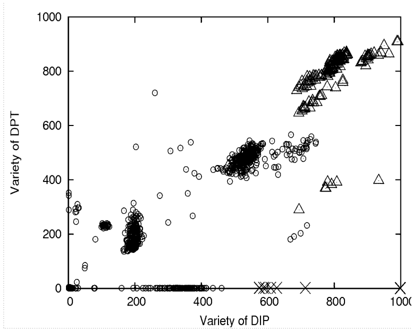

Figure 6. UDP packets

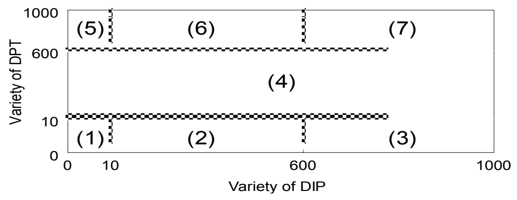

Figure 7. Seven Areas

Note that the analysis shown in this section follows an un-supervised learning framework. However, a simple frequent itemset mining algorithm can find problematic flows from actual internet traffic data. The analysis on the varieties of frequent itemsets contributes to the semi-automatic finding of problematic flows.

5. Discussion

Our team has manually analyzed the flows found by the proposed method in the previous section. However, the manual analysis is mainly aimed at confirming the feasibility of the proposed method. In most cases, network operators can use the results of the proposed methods to determine potential problems with the flows.

Table 4. UDP packets classification

Area		# of SIP/SPT Comb.	Collected Flows	Packets (%)
(1)	Unknown	69	1,313,263	44.75
(2)	DNS	31	112,958	3.85
	Unknown	4	7,073	0.24
(3)	Worm, Spam	22	103,670	3.53
(4)	DNS	26	583,501	19.88
	Broken Packets	2	11,003	0.38
	P2P	1	7,319	0.25
	Unknown	1	4,155	0.14
(5)	-	0	0	0.00
(6)	DNS	1	1,600	0.06
(7)		5	136,518	4.65
total		162	2,281,060	77.73

For example, scans and spam are plotted with crosses and P2P traffic with triangles in Figure 6. Clearly, there is a specific area for UDP packets where there are P2P and scan flows. The analysis of the UDP packets that share the same SIP and SPT can find potential problems. The analysis of TCP packets with the same SIP and SPT has a similar capability.

Other analysis using the proposed methods, such as the analysis of packets with the same SIP and DPT, the analysis of packets with the same DIP and SPT, and the analysis of packets with the same DIP and DPT, seems to be useful for analyzing client output behavior, client input behavior, and server input behavior, respectively. However, discreet investigation of these abilities remains a future research issue.

Note that the results shown in the previous section show one application area where a simple data mining technique can semi-automate analysis where previous attempts could not provide such a practical solution. Although only the number of varieties for 2-tuples and items are used, the analysis on the varieties of frequent itemsets is the key of this application.

6. Conclusion

This paper proposes a visualization technique of network status that uses a frequent itemset mining algorithm to find important phenomena in the network. The understanding of the ordinal network status and the rapid finding of security incident based on the proposed method help the naive users in monitoring and configuring network equipments.

A simple and one-pass stream mining algorithm which can simultaneously count the number of varieties in the found itemsets is also proposed. To show the effectiveness of the proposed algorithm, real internet traffic data was examined using the algorithm. The experimental results show that:

- The proposed algorithm can find P2P, Internet worms, and scans in actual Internet traffic.
- The performance of the proposed algorithm enables the on-line analysis of actual Internet backbones of up to 10-Gbps lines.

Since network management is an important issue in maintaining the Internet as an important social infrastructure, these results show an application area where a simple data mining technique can semi-automate analysis where previous attempts could not provide such a practical solution. Studying the best use of the proposed algorithm and the full use of its potential remain as future work.

Acknowledgements

The Internet traffic data analyzed in this study was provided by the WIDE project MAWI working group [4]. We would like to express our sincere thanks for their support.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in Proc. 20th Int. Conf. Very Large Data Bases, VLDB, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 12-15 1994, pp. 487-499.

[2] N. Jiang and L. Gruenwald, "Research issues in data stream association rule mining," SIGMOD Rec., vol. 35, no. 1, pp. 14-19, 2006.

[3] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Rfc2330, framework for ip performance metrics," 1998.

[4] Mawi WG, "http://www.wide.ad.jp/wg/mawi/," 2007.

[5] K. Yoshida, S. Katsuno, S. Ano, K. Yamazaki, and M. Tsuru, "Stream mining for network management," Transaction of the Institute of Electronics, Information and Communication Engineers, vol. E89-B, no. 6, pp. 1774-1780, 2006.

[6] E. D. Demaine, A. Lopez-Ortiz, and J. I. Munro, "Frequency estimation of internet packet streams with limited space." in In Proc. of the 10th Annual European Symposium on Algorithms, 2002.

[7] T. Mori, R. Kawahara, N. Kamiyama, K. Ishibashi, and T. Abe, "Detection of worm-infected hosts by communication pattern analysis," in Technical Report of the Institute of Electronics, Information and Communication Engineers, 2005 (in Japanese), pp. 1-6.

[8] M. Durand and P. Flajolet, "Loglog counting of large cardinalities," in European Symposium on Algorithms, ESA03, 2003.

[9] C. Estan, G. Varghese, and M. Fisk, "Bitmap algorithms for counting active flows on high speed links," in Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, 2003.

[10] A. Metwally, D. Agrawal, and A. E. Abbadi, "Efficient computation of frequent and top-k elements in data streams." in ICDT, 2005, pp. 398-412.