International Journal of Multimedia and Ubiquitous Engineering
Vol. 4, No. 2, April, 2009

ILR: a Self-Tuning Distributed Attacks Detection System

Wei Lin[1, 2], Liu Xiang[3], Derek Pao[2], and Bin Liu[1]
[1] *Department of Computer Science and Technology, Tsinghua University, China*
[2] *Department of Electronic Engineering, City University of Hong Kong, Hong Kong*
[3]*Department of Electronic Engineering, Tsinghua University, Beijing, China*
dr.linwei@gmail.com

Abstract

In order to protect Internet users from various attacks such as worms, viruses and other intrusions, signature-based intrusion detection system (IDS) should be deployed at the critical part of the network with rapid response for updating newly emerged attack signatures and containing the spread of worms or viruses at their early stage. The processing speed of one IDS cannot achieve the throughput requirement in the core networks because of the pattern matching, the key operation for signature-based IDS, is complex and time consuming. In this paper, it argues that if the signature set is shared by multiple IDSs, a packet needs to be checked once and once only by one of the IDSs, so traffic load can be redistributed among the IDSs to avoid local congestion. Packet marking is used to indicate the status of this packet utilized by collaborative IDSs, and a redistribution strategy named inner logical ring (ILR) is built among IDSs to redistribute the traffic load. Meanwhile, caching scheme is used to keep sequence for packets belonging to the same flow. This collaborative distributed IDS is robust with rapid response to various attacks, and the detection throughput is significantly increased from the throughput of the weakest IDS to the summation of all the collaborative IDSs.

1. Introduction

Malicious attacks frequently happen and cause millions of computers infected by various worms and viruses. Currently, only a part of enterprise users are protected by IDSs on the gateway of their networks, which are installed on the edge routers and maintained by network administrators belonging to different organizations. Other personal users install defending software on their computers and protect themselves on their own. It's quite difficult to keep all the computers with the most up to date defending software, because it is time consuming to update the signature database when new worms or viruses emerging for so many computers, which may lead to the wide spread of the worms or viruses and affect the normal Internet communication. Therefore, it would be a better choice if ISPs can deploy IDSs at the critical part of the core network, such as on the core routers, and provide users with services not only limited to the basic packet forwarding but also including intrusion detection and prevention. By doing so, IDSs in the core network can be updated rapidly and contain newly emerging attacks at their early stage within a local part of the network.

In order to find attacks hidden in network traffic, IDS on a router needs to compare the payload of every incoming packet with multiple patterns. This function is complex and time consuming. Since it's hard for software implementation to achieve high throughput, several

This work is supported by NSFC (60573121,60625201,60873250), the Cultivation Fund of the Key Scientific and Technical Innovation Project, MoE, China (705003), the Specialized Research Fund for the Doctoral Program of Higher Education of China (20060003058), 973 project(2007CB310701) and open project of state key Laboratory of Networking and Switching Technology (SKLNST-2008-1-05).

hardware-based methods have been proposed [1-4] with the average processing speed of several gigabits per second. Most current methods use the trick that matching hit rate is low in the network traffic, if not this case, such as worms break out on the Internet, the detection performance will degrade significantly. Furthermore, the link speed in the core network has reached terabits per second, if the IDS on a router checks every packet passing through, the throughput cannot scale to support such high speed and will be a bottleneck if the IDSs are deployed in the core network.

In this paper, a collaborative distributed IDS is proposed. It argues that if the signature set is shared by multiple IDSs, a packet needs to be checked once and once only by one of the IDSs. For that matter, packet marking is used to carry the indicating signal shared among inner IDSs, and inner logical ring (ILR) is built among routers to redistribute the traffic load to avoid local congestion. In addition, caching scheme is used to keep arriving sequence at the destination for packets belonging to the same flow. This collaborative distributed IDS is robust with rapid response to attacks and significant increase in detection throughput.

The rest of this paper is organized as follows. Related works will be presented in section II. The proposed distributed collaborative scheme will be illustrated in section III. Section IV is the performance evaluation and section V is the conclusion.

2. Related work

Much work has been done to do the attacks detection and containment in the network scope[5], which is in the categories of anomaly-based and signature-based schemes.

Anomaly-based schemes can detect unknown new worms. Since most networks behave in a particular and consistent fashion most of the time, when there are lots of abnormal phenomena occurring on the network, it often means something is wrong.

Anomaly-based detection schemes require the definition of normal network behavior. The big challenge of anomaly-based schemes is defining what normal network behavior is, deciding the threshold to trigger the alarm, and preventing false alarms. The users of the network are normally human, and people are hard to predict. If the normal model is not defined carefully, there will be lots of false alarms, and the detection system will degrade performance.

Signature-based schemes compare the payload of every packet with attack signatures in database. One big challenge of the signature-based IDS is that every signature requires an entry in the database, so a complete database might contain thousands of entries. Each packet is compared to all entries in the database, which can be very resource-consuming and doing so will slow down the throughput, make the IDS vulnerable to DoS attacks. Some IDS evasion tools use this vulnerability and flood signature-based IDSs with too many packets to the point that the IDS cannot keep up with traffic, thus making the IDS time out and miss packets, and as a result, possibly miss attacks.[6]

Wong et al.[7] showed that deploying IDS at backbone routers is as effective as deploying at all the hosts the backbone routers cover. But it is more complex to install IDSs at every single host than only at backbone routers.

Most IDSs are designed for local area or enterprise network detection and containment. But ISPs have a greater position in worm containment since they are the junction of data exchange. In [8], Moore et al. showed that implementing containment in the ISP ASs is far more effective than in the customer's AS. After detecting worms from a certain customer network, the ISP can slow down or block off partial traffic from that network to prevent worms from spreading to other customer networks. The detection might be more complex for signature-based schemes because of dealing with large amounts of traffic.

3. Collaborative distributed scheme

If the pattern sets of different IDSs run by an ISP are the same, it will be unnecessary to check a packet multiple times. Since router is normally stateless, it will never know whether an incoming packet has been checked or not unless the checking information can be extracted. And redundant checking can be eliminated if the checking information is available and authentic. Therefore, collaborative mechanism can be built to share the traffic load among IDSs on routers for the purpose of alleviating the workload pressure for each IDS.

For the ease of discussion, in this paper, the pattern set of each router belonging to the same ISP is assumed to be the same. If not the case, a common pattern set can be derived and compared only once with each packet forwarded by routers of the same ISP, and the unique patterns can be compared with separately by each router.

In order to get the checking status information of a packet in real time, two unused bits in the packet header are used by the collaborative IDSs as indicators to indicate the checking and routing status of this packet, which will be discussed later.

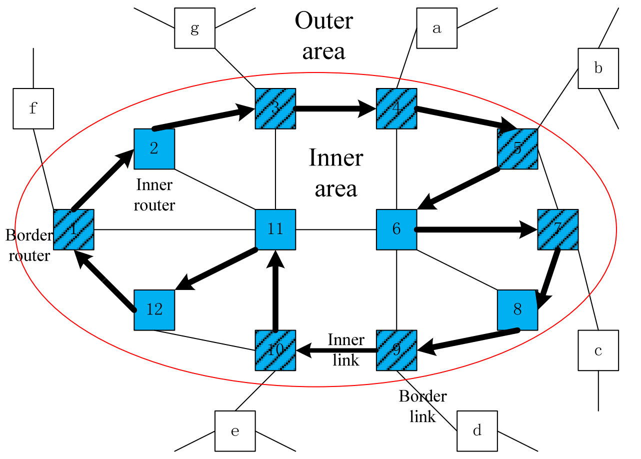

Figure 1. Routers sharing the same rule set form the inner area

Even if checking information indicated by the marking bits is available, it can be not true which may be modified and sent from a malicious user to confuse the detection system. To make the collaborative detection mechanism reliable, reliability among routers is limited within a scope. Currently, the most practical reliable scope is interconnected routers in the same ISP AS, where routers can be configured by the same pattern set and follow the same strategy.

As shown in Fig.1, with the boundary, the whole network is logically divided into two parts from the ISP's view, reliable inner area and unreliable outer area. The rest routers and hosts form the latter.

In the following, the problem will be defined formally.

The number of inner area routers is N, and the name of each router is Ri_n where n is an integer ranging from 1 to N. The capability to do attack detection for Ri_n is Ai_n. The sum of the capability to do attacks detection of the N routers is $Ai_{sum}(N)$

$$Ai_{sum}(N) = \sum_{n=1}^{N} Ai_n$$

Assuming the connection type between inner routers with outer area nodes is point to point, the number of inner routers' physical ports connected with outer area nodes is M, and the name of each port is Po_m where m is an integer and ranges from 1 to M. The amount of traffic coming from outer area into the inner area through Po_m is Ao_m. The sum of the amount of traffic coming into the inner area through the M ports is $Ao_{sum}(M)$.

$$Ao_{sum}(M) = \sum_{m=1}^{M} Ao_m$$

Considering the current single router detection methods, the bottleneck of the system is the router with the weakest capability to do attack detection. The system will be overflowed, when $Ao_{sum}(M) > MIN(Ai_n)$ and all the traffic passes through the router with weakest capability to do attack detection.

The optimum solution is the system can work properly when $Ao_{sum}(M) < Ai_{sum}(N)$. Since network congestion usually happens in local part of the network and mainly caused by non-uniform traffic distribution, some routers are heavily loaded while others are still with light load. To handle this, collaborative distributed attacks detection strategy can be used to rebalance the traffic among routers. As overall network congestion rarely happens and actual network traffic is a small portion of the traffic in the worst case, it almost always meets the condition that $Ao_{sum}(M) \leq Ai_{sum}(N)$.

In this paper, we propose a collaborative distributed intrusion detection scheme called inner logical ring (ILR) to increase the overall checking throughput, accompanied by an LRU-based cache scheme to quarantine flow sequence.

In order to achieve the above goal, ILR has three aspects to be solved.

The checking status of a packet should be shared among inner routers in real time, which is implemented by packet marking scheme. Normal packets are marked as 'checked' state and malicious packets are simply discarded.

A packet should be checked by one inner router before it leaves the inner area. If routers on the shortest path are fully occupied, other inner routers on the redistribution path, known as ILR, are employed to help checking this packet.

Flow sequence should be guaranteed by LRU-based cache scheme while packets are redistributed among inner routers. So, whether a packet should be checked by the router will be instructed according to the recently arrived flows in the cache.

As shown in Fig. 1, routers which share the same set of attacks detection rules form a connected graph, and these routers are defined as inner routers. Inner routers and links between any two inner routers form the inner area, while other routers, hosts and links form the outer area. If an inner router has a port connects with outer area, the router is defined as border router and the port is defined as border port. Inner routers are connected by some inner links in thick solid lines to form an inner logical ring. Each inner router has a logical neighbor on the clockwise of the ring. If a packet cannot be checked by routers on the shortest path, it will be sent to the clockwise neighbor router on the logical ring, until one router have the capability to check this packet. Then the packet will be sent to destination along the shortest path from this router.

We use two undefined bits in the packet header to indicate the status of the packet. Information exchanging among inner routers is thought to be reliable, while information coming from outer area is considered to be unreliable. In order to avoid cheat, both the two bits are set to 0 by the border router with diagonal lines in Fig. 1, when the packet enters into the inner area.

C bit: it's used to instruct the checking strategy of IDSs in the inner area. 0 represents this packet has not yet been checked in the inner area. While 1 represents this packet has been verified a normal one to be forwarded to the destination. Malicious packet will be directly discarded by the router checking it without further forwarding.

R bit: it's used to instruct the forwarding strategy of routers in the inner area. 0 represents this packet is normally forwarded along the shortest routing path while 1 represents this packet is in special routing mode, which will be described in details later.

Packet marking is most easily done by using the DSCP field in the IP header. If this is not possible, other packet marking strategies like IPv6 extension headers or using a private address space and tunneling can be imagined[9].

The most efficient way is all the packets can be checked on the shortest routing path before leaving the inner area, for they can be forwarded by the least routers on the way to destination, without adding extra forwarding workload to the inner routers. The entrance router is supposed to process incoming packet if it has spare capacity. Otherwise, the packet will be forwarded along the shortest path until one router can handle it. However, when the traffic load is heavy, not all the packets can be checked by the fully-occupied routers on the shortest path. To solve this problem, we use links to connect the inner routers as a logical ring, defined as Inner Logical Ring (ILR), to redistribute the traffic when part of the inner routers don't have additional capabilities to check the newly arrived packets. Since the routing throughput is much higher than the pattern matching throughput, traffic can be sent to any node quickly to do pattern matching. As long as $Ao_{sum}(M) \leq Ai_{sum}(N)$ is true, the system can work properly.

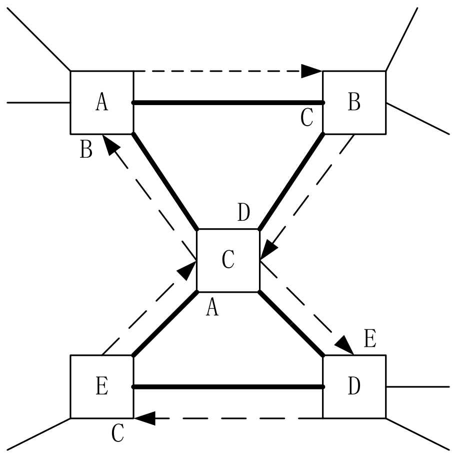

Figure 2. Router C has two logical positions on the ring

To form the logical ring to connect all the inner routers in series, in some cases a router may have more than one logical position on the ring, which means the ring will pass the router more than once from different physical ports. Each logical position will be assigned an outgoing port number to its clockwise neighbor. This information will be bound to the incoming port. A mapping table (MT) can be used to record the incoming port and the corresponding outgoing port. With the help of MT, each router will know the route to its clockwise neighbor on the ILR. As shown in Fig. 2, inner routers form a logical ring in the sequence of <A, B, C, D, E, C, A>. Router C has more than one logical position on the ring, and C has two logical neighbors, which are D and A. Usually, routers in the core network are nearly fully connected, by carefully constructing the logical ring, each router can have only one position on the ring.

ILR will not function until the inner routers along the forwarding path cannot handle all the packets. In this case, the last stage router within the inner area will mark R bit of the unchecked packet as 1 and route it to the ILR neighbor recursively until one has the capability to process this packet. If the last stage router has more than one position on the ring, its clockwise neighbor is assigned according to the hashing value of the flow ID, so that packets of the same flow will be sent to the same logical neighbor. For router having multiple logical positions and without additional processing capability, packets with R bit equal to 1 coming from one of its anticlockwise neighbors will be sent to the corresponding clockwise neighbor according to the MT. Once checked, R bit of this packet will be marked as 0, which means this packet resumes to normal routing state and it will be forwarded to the destination along the shortest path according to the routing table. During a packet being forwarded around the ILR, its TTL field will not be changed until it resumes to normal routing state.

Due to the limitation of pattern matching capability of one router, packets may be distributed to different inner routers to do the pattern matching task. Because of the different processing speed and varying queuing lengths, packets in the collaborative system may not be forwarded in a FIFO mode, resulting in disorder problem. Though it's OK for UDP packets and TCP packets belonging to different flows, it will be a problem for TCP packets of the same flow if packets out-of order occurs seriously. Hashing flows rigidly to a particular router to do pattern matching would be a choice, but since the incoming traffic cannot be hashed uniformly among inner routers, and the selected router may not on the routing path, the overall system robustness cannot achieve optimum.

In this paper, the balance between system robustness and keeping flow sequence is solved by the observation that if the arrival interval between two consecutive packets is long enough, they can be processed by different routers and go along different paths with very low probability of disorder, which can be handled by TCP glide window without affecting the application-level performance.

An LRU-based cache is used to record recently arrived flows. When the cache is full, the flow A which is the least recently used in the cache will be replaced by the new incoming flow. So flow A will be treated as a new flow when it arrives later.

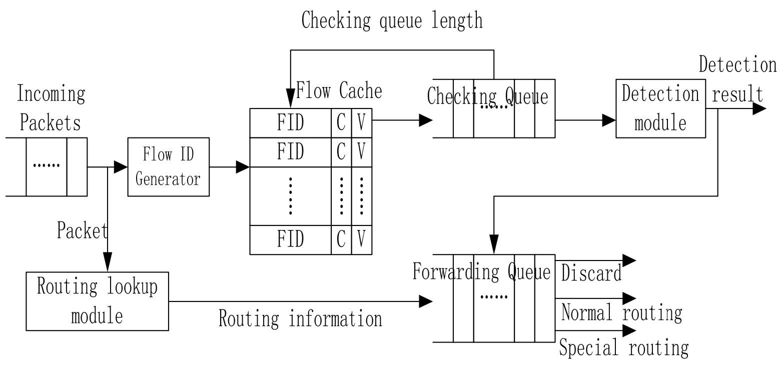

Figure 3. Architecture of IDS in a router

As shown in Fig. 3, all the incoming packets read out from the incoming FIFO will be sent to the routing lookup module to get the next hop forwarding information, but only the unchecked ones can be sent to the flow ID generator module, which uses the five-tuple (source and destination IP address, source and destination port number, protocol type) as input and generates 32 bits flow ID in CRC-32 method.

There are three fields in the cache, FID is used to record the flow ID, C bit is used to record packets of this flow should or shouldn't be checked by this router, and V bit is used to indicate whether or not this entry is valid. The cache can be implemented by BCAM and the LRU relationship can be recorded by a link list. The input flow ID is compared with valid entries in cache. If there is a cache hit and the C bit of the matched entry is equal to 1, the packet is sent to the checking queue and the matched entry is moved to the head of the link list; If there is a cache hit and the C bit of the matched entry is equal to 0, the packet is sent to the forwarding queue and the matched entry is moved to the head of the link list; If there is a cache miss and the length of the checking queue doesn't exceed the almost full threshold (3/4 of the maximum length), the packet will be sent to the checking queue and a new entry is saved in the cache in LRU manner with C bit equal to 1; If there is a cache miss and the length of the checking queue does exceed the almost full threshold, the packet will not be sent to the forwarding queue, and a new entry will be saved in the cache in LRU manner with C bit equal to 0.

The detection module reads one packet from the checking queue and checks the packet with attack signatures in the pattern set. According to the checking result, if it's a normal packet, the C bit of the packet is set to 1 and the packet is sent to forwarding module, otherwise, it is discarded.

In this way, every packet is checked by one and only one IDS in the inner area. IDSs on the packet's forwarding path have higher priority to be selected to check the packet. However, if they are heavily loaded and cannot handle more packets, IDSs on the ILR are sequentially selected until one can process this packet. Then normal packet is forwarded to its destination along the shortest path whereas malicious packet is simply discarded. In order to keep the flow sequence, LRU-based cache is used for each IDS to record the recently arrived flows. Packets of the same flow are usually forwarded along the same path unless the arrival interval is large enough for packets of the same flow can go along different path with low disorder probability, which can be easily handled by sliding window.

4. Performance evaluation

Author names and affiliations are to be centered beneath the title and printed in Times New Roman 12-point, non-boldface type. Multiple authors may be shown in a two or three-column format, with their affiliations below their respective names. Affiliations are centered below each author name, italicized, not bold. Include e-mail addresses if possible. Follow the author information by two blank lines before main text.

According to the statistical result, the number of parallel packet flows in a core-router in CERNET was reaching 300 million in 2005[10]. Internet traffic is dominated by TCP flows and UDP flows[11]. In this paper, UDP packets of the same flow can be checked at different IDSs to balance the traffic load, given that UDP packets don't need to be kept sequential.

The network topology adopted in simulation is built using the NW small-world network model[12] in two steps: 1) start from a regular nearest-neighboring network of N nodes (N=100), in which node is connected to its 2K neighbors (K=1); 2) at every step, with probability p (p=0.2), add an edge to the network. And the border ports which are connected to the outer area are generated randomly. The average number of hosts covered by one backbone router is about 100.

Then we use the Shortest-Distance-Vector algorithm to build a routing table for each router. After that we generated the input packets according to the traffic distribution in [11].

The total number of flows in the simulations is 10000, and the simulation time lasts 10000 cycles.

The differences between stand-alone method and our collaborative method are as follows: 1) stand-alone method checks each packet only at the edge routers. In this way, if a packet cannot be checked by the entrance or exit edge routers, it will arrive at the destination without detection. 2) Our collaborative method checks each packet only once on its way to the destination. All the routers have the capability of intrusion detection. If a packet remains unchecked when it arrives at the exit border router the first time, it will be sent to its ILR neighbor recursively and its R bit will be set to 1 until one router has the capability to check this packet. After that, it will be forwarded to the destination along the shortest path. And the packet will not be discarded unless it arrives at the destination the second time with C bit equals to 0. In order to deal with the disorder problem, LRU manner was used to handle different cases.

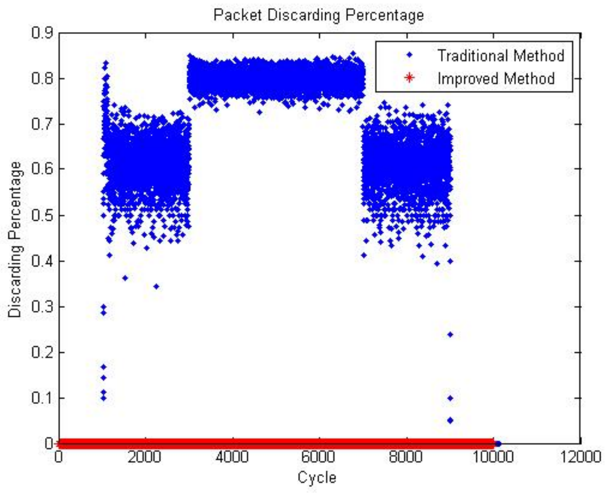

Figure 4. Packet Discarding Percentage

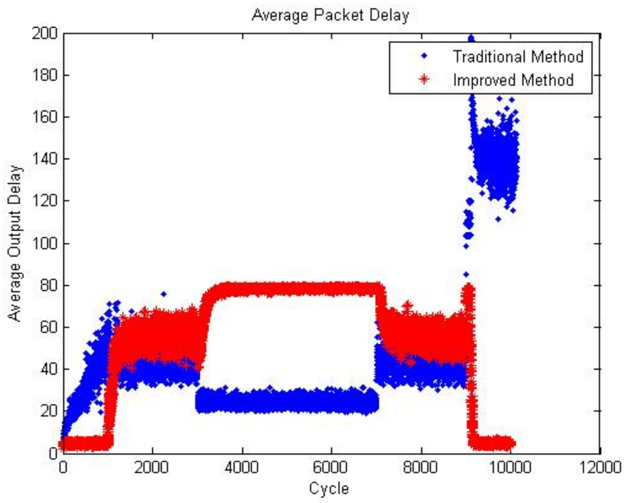

Figure 5. Average Packet Delay

As shown in Fig. 4 and Fig. 5, traffic load is varied from light to heavy then back to light, and the heaviest traffic in a simulation cycle amounts to $Ai_{sum}(100)$, the sum of the capability to do attacks detection of the 100 routers. During the whole simulation cycles, our collaborative method hasn't discarded a single packet, which effectively proofs that as long as $Ao_{sum}(M) \leq Ai_{sum}(N)$ is true, the system can work properly. When the traffic is light, there's no packet discarded in both methods, but the average packet delay of stand-alone method increases rapidly while ours maintains to be fairly short. Although the average packet delay of our collaborative method increased relatively high whereas that of stand-alone method decreased a lot as the traffic load aggravating, our method is still robust without discarding any packet. However, in contrast, a majority of packets have been discarded in stand-alone method since the unbalanced load distribution and traffic congestion. And its decreasing packet delay is a feint since lots of long-delay packets have been discarded in the midway.

The simulation result shows that when the deployment ratio is 1%, the collaborative method is as effective as 50% hosts are deployed with IDSs. The detection throughput is the summation of all the routers' detection capability, which is independent of the traffic distribution. When the traffic load is light, the delay is almost the same as the stand-alone method. When the traffic load is heavy, the delay increases but it still can check all the incoming packets, while the stand-alone method has already missed detection of some packets. Flow sequence is guaranteed; about 2.3% packets are out of order, which can be handled by the sliding window at the destination.

5. Conclusion

In this paper, a collaborative distributed intrusion detection method is proposed, which can be deployed on the backbone routers in the same ISP AS. In order to fully utilize the detection capability and increase the overall detection throughput, a packet only needs to be checked once in one ISP AS along its way to the destination on the premise that the attack signatures can be shared by the routers of the same ISP. For details, three specific techniques have been employed. Packet marking is used to share the detection and routing status. Inner logical ring is built to redistribute the packet when none of the routers on the shortest path has the capability to check it. LRU-based cache in each router is used to keep TCP flow sequence while UDP packets can be checked at any router to balance the work load. The simulation result shows that the overall detection throughput increased significantly since the adoption of our collaborative method, and the deployment ratio is much less than that deploying IDSs at the hosts.

References

[1] Derek Pao, Wei Lin, and Bin Liu, Pipelined Architecture for Multi-String Matching. Computer Architecture Letters, 2008.
[2] Tian Song, et al., A Memory Efficient Multiple Pattern Matching Architecture for Network Security, in IEEE INFOCOM 2008. 2008.
[3] Jan Van Lunteren. High-performance pattern-matching for intrusion detection. INFOCOM 2006. Barcelona, Spain.
[4] Rong-Tai Liu, et al. FTSE: The FNP-like TCAM searching engine. ISCC 2005, p 863-868, Murcia, Spain.
[5] Pele Li Mehdi Salour, Xiao Su, A Survey of Internet Worm Detection and Containment. IEEE Communications Surveys, 2008. 10(1).
[6] R.A. et al, Snort 2.1 Intrusion Detection, 2nd ed.: Syngress, O' Reilly, 2004, pp. 490-491.
[7] Cynthia Wong, et al. Dynamic quarantine of Internet worms. in the proceedings of the International Conference on Dependable Systems and Networks, 2004. Florence, Italy.
[8] David Moore, et al. Internet quarantine: Requirements for containing self-propagating code. INFOCOM 2003. San Francisco, CA, United States.
[9] Amund Kvalbein, et al. Fast IP network recovery using multiple routing configurations. INFOCOM 2006. Barcelona, Spain.

[10] Guang Cheng, et al., Hash algorithm for IP flow measurement. Ruan Jian Xue Bao/Journal of Software, 2005. 16(5): p. 652-658.
[11] Kevin Thompson, Gregory J. Miller, and Rick Wilder, Wide-area internet traffic patterns and characteristics. IEEE Network, 1997. 11(6): p. 10-23.
[12] Newman M E J and Watts D J, Renormalization group analysis of the small-world network model. Phys. Lett. A, 1999. 263: p. 341-346.

Authors

Wei Lin received his bachelor degree in the Department of Computer Science and Technology, Dalian University of Technology, China in 2004. Then he was recommended to pursue his doctor degree directly in the Department of Computer Science and Technology, Tsinghua University, China. He was selected as a joint PhD student in department of Electronic Engineering, City University of Hong Kong in 2006. His research interests are route lookup, flow classification and deep packet inspection.

Liu Xiang is an undergraduate student in the Department of Electronic Engineering, Tsinghua University, China. She participates in a Student-Research-Training Program in the Lab of Broadband Network Switching Technology and Communication, Department of Computer Science and Technology, Tsinghua University, China.

Derek Pao received his B.Sc. (Eng.) degree in Electrical Engineering from the University of Hong Kong in 1984. After working for RCL Semiconductors Ltd. as an Integrated Circuits Design Engineer for one-and-a-half years, he pursued higher education at the Concordia University, Montreal, Canada, where he obtained his master and Ph.D. degrees, both in Computer Science, in 1987 and 1991, respectively. Before joining the Electronic Engineering Department of the City University of Hong Kong in 1992, he was an Assistant Professor of Computer Science at the Concordia University.

Bin Liu got the Bachelor, Master and PhD degree of Computer Science from Northwestern Polytechnical University, Xi'an, China in 1985, 1988 and 1993 respectively. From 1993-1995 he worked in Beijing University of Posts and Telecommunications as a postdoctoral fellow. In 1995 he became as an associated professor at Tsinghua University. He become a full-professor in 1999 and was promoted as a Ph. D supervisor in the following year. He likes running, mountain-climbing, swimming, playing volleyball, ping pang, and Chinese chess. He prefers noodles, gruel, boiled dumplings and some sweet food. He is open-mind, patient, humorous, and easy-going.