

Short Paper

A Distributed Intrusion Detection Model for the Domain Name System

CHANG-SHENG CHEN, SHIAN-SHYONG TSENG
AND CHIEN-LIANG LIU

*Department of Computer Science and Information Science
National Chiao Tung University
Hsinchu, 300 Taiwan*

We have investigated the problem of detecting DoS-like DNS anomalies in DNS system. In this paper, we propose a distributed Two-phase DNS anomaly detection model for solving the problem. Three sets of algorithms corresponding to different configurations are proposed, including one sequential algorithm and two distributed algorithms, each with an increasing level of parallelism. The complexity of these algorithms have been found to be $O(n \log n)$. The distributed algorithms show at least a constant $(1-1/C_k)$, $C_k > 1$, improvement over the sequential one. To evaluate the performance, we have implemented the algorithms and applied them to a number of examples. The experimental result shows a speed up of about 1.68 on the test example for running on an enhanced distributed architecture with C-IDS over the sequential one. A higher speedup might be common because DNS anomalies will make the traffic distribution more concentrated on the outliers, and the computation will usually converge much more quickly.

Keywords: DoS, DNS, distributed two-phase DNS anomaly detection, IDS, two-phase anomaly detection algorithms

1. INTRODUCTION

The Domain Name System (DNS) [1, 2] is an essential part of the Internet infrastructure. Given the importance of DNS servers, direct or indirect attacks on them are common. In this paper, we would like to focus our study on developing a distributed detection model for DoS-like (or DDoS-like) [3-5] attacks to the DNS (i.e. all that might cause too many queries in a short specific interval). Other issues [6, 7] (such as DNS-spoofing, DNS server exploits, etc) are not covered here. We propose a distributed Two-phase DNS anomaly detection model for solving the problem. First, in the outlier detection phase, a radix (patricia) [8, 9] tree based aggregation profiler is adopted to detect outliers in the incoming DNS traffic. Second, DNS anomaly identification phase is proposed to identify candidates of the problem types and the real sources (that induce the anomalies). Furthermore, to further reduce the computation, a new subsystem with C-IDS (database Center for sharing identified IDS information) and an encod-

Received September 13, 2001; accepted April 15, 2002.

Communicated by Jang-Ping Sheu, Myongsoon Park and Makoto Takizawa.

ing scheme (for encoding newly identified DNS anomalies and implemented as specialized DNS zones) is introduced. Three sets of algorithms are proposed, including one sequential algorithm and two distributed algorithms, each with increasing levels of parallelism. The complexity of these algorithms have been found to be $O(n \log n)$. The distributed algorithms show at least a constant $(1-1/C_k)$, $C_k > 1$, improvement over the sequential one. To evaluate the performance, we have implemented the algorithms and applied them to a number of examples. By analyzing our experimental results, we have shown that the whole system can become an effective means of detecting and defending against DNS network attacks.

2. BACKGROUND

In general, there are anomalies located in the midst of normal DNS activities. For example, the 3rd and 5th lines in Table 1 (as shown below) are packets of bogus DNS queries [7, 10]. We need a system for identifying the problem types and the real sources. By analyzing traffic data (i.e. *tcpdump* [11] dump trace) collected in close temporal proximity (e.g. 10000 continuous DNS packets), it is expected that the system finish the computation job in a short interval (e.g. in 10 minutes).

Table 1. Sample from a typical tcpdump trace dumped file.

Sample Entries from a typical trace						
1).	20:01:09.479863	10.113.146.21.34103	>	10.113.1.1.53:	57873+	A?
		pchome.com.tw. (31) (DF) (ttl 253, id 11362)				
2).	20:01:09.480741	10.113.1.1.53	>	10.113.146.21.34103:	57873	1/4/0
		pchome.com.tw. A 210.59.230.60 (123) (ttl 64, id 24644)				
3).	20:01:10.716969	10.113.156.191.137	>	10.113.1.1.53:	41101+	A?
		97.186.111.63.xyz.com. (44) (ttl 126, id 43349)				
4).	20:01:11.190273	10.113.23.5.1560	>	10.113.1.1.53:	25196+	PTR?
		131.200.168.192.in-addr.arpa. (45) (ttl 62, id 19429)				
5).	20:01:11.356617	10.113.208.131.137	>	10.113.1.1.53:	62793+	A? 10.230.13.37.
		(32) (ttl 122, id 54324)				

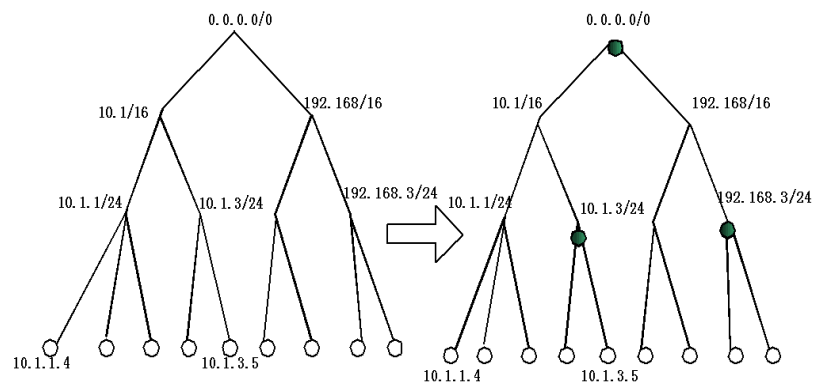


Fig. 1. Aggregation profiler concepts: small entries are aggregated aggregates.

In general, it is still not known how to design or implement a system for solving DoS-like problems effectively. However, some implementations have provided promising results. For example, Aguri [8], is an aggregation-based traffic profiler (as shown in Fig. 1) targeted for near real-time traffic monitoring and had been used for some cases of DoS detection successfully. According to domain expertise, normally over 99% of the incoming DNS packets use UDP. With a little modification, we could use similar DNS UDP traffic aggregation-based approach for conducting DNS outlier detection (i.e. focusing on incoming traffic to the DNS with a destination UDP port = 53 only).

3. SYSTEM MODEL AND ARCHITECTURE

As shown in Fig. 2 (a) and (b), the proposed model for a general distributed Two-phase DNS anomaly detection system is composed of a group of independent distributed IDS's and a specialized C-IDS (i.e. database Center for sharing identified IDS information). Each of the distributed IDS is mainly composed of two components: an outlier detection engine (located in each DNS system) and an anomaly identification engine (located in another host nearby, for releasing the computational burden on each DNS). By doing traffic aggregation, the Phase-1 subsystem (as shown in Fig. 3a) could detect if there is outstanding abnormal DNS traffic in a specified interval. Then, the outliers are disabled from accessing the DNS. Furthermore, for reducing the computational burden, a new C-IDS check (i.e. DNS query) will be introduced. If the anomaly had been reported, the Phase 2 (as shown in Fig. 3b) will be skipped and back to Phase 1. On the other hand, after receiving a trigger signal, the Phase-2 subsystem will initiate a series of pattern matching process for identifying candidates of the problem types (as shown in Fig. 4 below) and the real sources. After finishing Phase 2, a new sub-step for conducting DNS encoding (e.g. example entries as shown in Table 2) and C-IDS database update is introduced (i.e. once some anomalies identified by any independent IDS, the related information will be pushed on the C-IDS database for sharing among the IDS groups). After that, the system repeats the Phase-1 process.

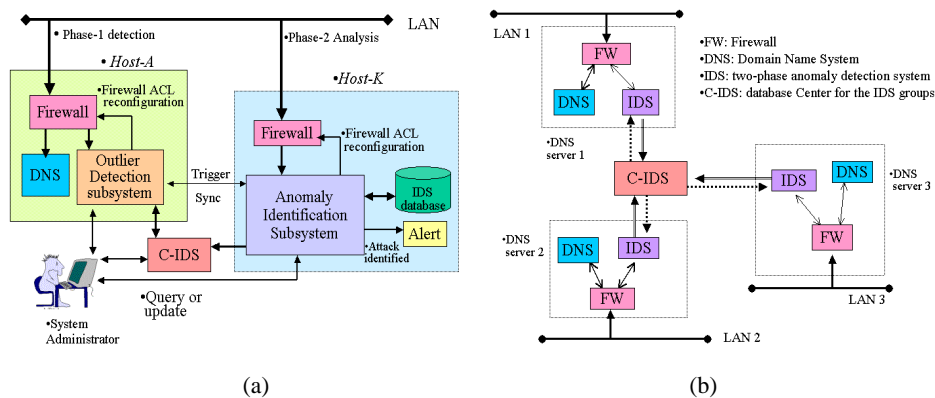


Fig. 2. The proposed model for a general distributed two-phase DNS anomaly detection.

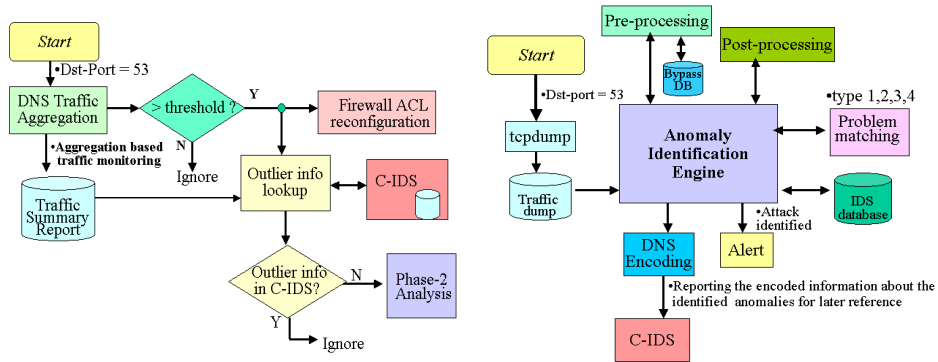


Fig. 3. (a) Phase-1 outlier detection (b) Phase-2 anomaly identification.

Table 2. Explanation about the DNS encoding of a typical C-IDS entry.

Item	Function	Description	Field
1.	Pseudo-network	127.0.0.0	1 st field
2.	Problem type	Type 1 => 10, Type 2 => 20, Type 3 => 30, Type 4 => 40, etc	2 nd field
3.	Problem instance	1,2,3,etc.	3 rd field
4.	DNS server	One of the server list (e.g. {1,2,3})	4 th field

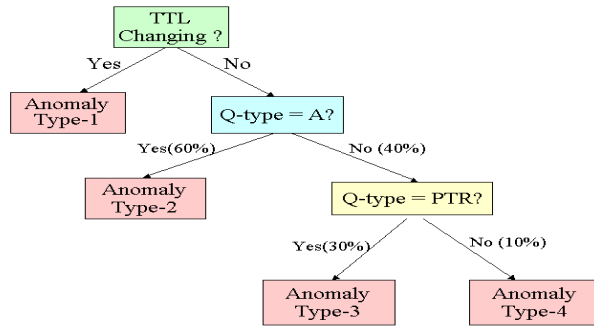


Fig. 4. Our classification scheme of the DNS anomalies.

4. ALGORITHMS

In all cases, the Phase-1 outlier detection should be done for locating outliers. If outstanding outliers are located, another Phase-2 process will be followed for anomaly identification. In this section, three sets of algorithms (corresponding to different configurations) are proposed, including one sequential algorithm and two distributed algorithms, each with increasing levels of parallelism. For the distributed algorithms, the synchronization among the Phase-1 and Phase-2 processes of each IDS and the C-IDS are done mainly using the DNS framework and the Remote Procedure Call mechanism.

4.1 Data Structure for the DNS Anomaly Detection Phase

For conducting the detection, we have extracted some specific attributes (such as source IP address, DNS query type, query instance) from the incoming DNS packet streams and encoded them in the following algorithms. Each encoded entry is of fixed length and will have a generic form as shown in Fig. 5 below. There will be 9 fields (field 1 for indicating query-type, fields 2 to 5 for indicating source ip address, fields 6 to 9 for indicating query instance). In this way, the Phase-2 anomaly identification process is converted into a “general sorting and counting” problem.

Encoded Query-type	Encoded Source ip address	Encoded Query instance
--------------------	---------------------------	------------------------

Fig. 5. The generic format for an encoded entry.

- The query-type will be converted to a number (1, 2, etc.).
- The source IP address will be put into 4 fields of 3 digits, patching leading zero when necessary
- For the reverse PTR cases, the numbers of the 4 octets of a PTR query will be extracted and put into fields 5 to 9. However, for forward cases, since the length is variable, we will only use the left 12 characters, including the dot delimiter, encoded into the number 0. The approximation is reasonable since when the Phase-2 anomaly identification is conducted, there must be lots of the same outliers detected, and so, the traffic will show more concentration.

For example, here are some instances of the attributes extracted from typical DNS traffic trace. After encoding, the corresponding entries may look the following (as shown in Fig. 6):

1	192	168	001	002	www	0pc	hom	e0c
1	010	113	034	056	www	0mi	cro	sof
2	192	168	034	123	121	045	067	010
1	192	168	001	002	www	0pc	hom	e0c
2	192	168	034	123	121	045	067	010

- Field 1: query-type (1, 2, 3, 4)
- Fields 2-5: encoding of source IP address
- Fields 6-9: encoding of query instance

Fig. 6. Sample encoded entries.

4.2 The Sequential Approach

In this subsection, we propose a sequential algorithm. The Phase-1 outlier detection was conducted in Step1 and Step2. Next, the remaining steps perform the Phase-2 anomaly identification analysis.

• **Sequential algorithm for a single DNS server**

Input: The incoming DNS query packets.

Output: DNS traffic summary reports, Identified DNS anomalies and e-mail notices.

Step1. Conduct the DNS traffic aggregation and periodically generate summary reports.

Step2. Periodically check the aggregation summary report. If there is no entry greater than the threshold (e.g., 1%), then go back to Step1.

Step3. Initiate a DNS trace dump (e.g., 10000 packets).

Step4. Conduct data pre-processing. (See the Procedure listed below)

Step5. Conduct pattern matching for identifying DNS anomalies.

Step6. IDS database update. Store the newly identified anomalies in the IDS database.

Step7. Merge the outliers' IP into the access control database of the firewall. (disabled from access)

Step8. Generate a System Alert (i.e., by e-mail). Next, go back to Step1.

• **Procedure for data preprocessing**

Input: The incoming DNS query packets.

Output: The classified data for DNS anomaly identification

Step1. Ignore the IP from the bypass database (e.g., from normal DNS/mail/proxy server, etc.).

Step2. Classify the queries according to DNS anomaly types for later computation.

• **Analysis of the complexity of the sequential algorithm:**

The total computation time is described by

$$\text{Seq}(n) = T_{p1}(n) + T_{p2}(n) \quad (\text{EQ-5.1})$$

where, $T_{p1}(n)$ = Running time of Phase-1 outlier detection,

$T_{p2}(n)$ = Running time of Phase-2 anomaly identification.

The sets of source IP address and the DNS query are the two relevant factors. In Phase-1 (Step1), only the source IP and the related packets counts are considered. We use a radix (patricia) tree based aggregation program (adopted from aguri [12]) to detect outliers in the incoming DNS traffic. Assume that there are n DNS packets to analyze. Since both the complexity of radix (patricia) tree sorting for IP addresses and post-order tree walk are time linear [7, 8], the complexity of the phase-1 in all the configurations is $O(m)$. On the other hand, the pattern matching procedure (Step5) dominates of the entire Phase-2 process. All the other steps (i.e., Step3-8, with Step5 excluded) can done in one pass. From subsection 4.1, we find that the Phase-2 process has been converted into a process consisting mainly of sorting a fixed-length table of entries and counts. In general, the sorting will be finished in $O(n \log n)$, while the related counting can be done in $O(n)$. Thus, the complexity of the Phase-2 process, and hence the entire process is $O(n \log n)$.

4.3 A Simple Distributed Two-Phase Approach

In this subsection, we propose a simple distributed algorithm that partitions the jobs into two parts. The Phase-1 subsystem and the Phase-2 subsystem will be conducted on

different hosts. Here is the common data structure for synchronizing the Phase-1 and Phase-2 processes.

```

Structure ids-data {
    Algo-mode: integer; Prob-type: integer; Call-phase2: boolean; Anomaly-found:
    boolean;
} IDS;

```

• **A simple distributed two-phase algorithm**

Input: The incoming DNS query packets.

Output: DNS traffic summary reports, Identified DNS anomalies (filed in the local IDS database) and e-mail notices to people for fixing the problem.

Step1. Conduct Phase-1 DNS outlier detection. If the returned result **IDS.Call_Phase2** = FALSE, repeat Step1.

Step2. For each DNS anomaly type, conduct Phase-2 analysis (**IDS.Algo-mode** = Simple-distributed, **IDS.Anomaly-type** = 1,2,3,4).

Step3. Join the results of each type.

Step4. Go back to Step1.

• **DNS outlier detection algorithm (common to all configurations)**

Input: **IDS.Algo-mode**

Output: The result **IDS.Call-phase2** (TRUE or FALSE,)

Step1. Conduct DNS traffic aggregation and periodically generate summary reports.

Step2. Set **IDS.Call-phase2** = FALSE. Check the newly generated aggregation summary report. If there is no outlier entry greater than the threshold (1%), go to Step6.

Step3. Set **IDS.Call-phase2** = TRUE. Merge the outliers' IP into the access control database. The outliers are disabled from accessing the DNS.

Step4. If **IDS.Algo-mode** = Simple-distributed, go to Step6.

Step5. Consult the C-IDS database. If there is information about the newly identified outliers, set **IDS.Call_Phase2** = FALSE.

Step6. Return the result **IDS.CALL_Phase2**.

• **Phase-2 Algorithm for distributed DNS Anomaly Identification**

Input: **IDS.Algo-mode**, **IDS.Prob-type**

Output: The result **IDS.Anomaly-found** (TRUE or FALSE).

Step1. Check for the incoming trigger signal from the main program. If no trigger signal (e.g., using e-mail NOTICE, or Remote Procedure Call) is received, repeat Step1.

Step2. Initiate an online DNS traffic trace dump (e.g., 5000 packets).

Step3. Set **IDS.Anomaly-found** = TRUE

Step4. Conduct Data Pre-processing according to the input **IDS.Prob-type**

Step5. Conduct pattern matching for identifying DNS anomalies. If no DNS anomalies are identified (i.e., over the minimum support), set **IDS.Anomaly-found** = FALSE and go to Step7.

Step6. Conduct DNS-encoding, local IDS and C-IDS database update

Step6.1 Store the newly identified anomalies in the local IDS database.

Step6.2 If input variable **IDS.Algo-mode** is Simple-distributed, go to Step 6.

Step6.3 Encode the identified anomalies.

Step6.4 Store the newly generated information in the C-IDS database via DNS dynamic update.

Step7. Generate System Alert by e-mail.

Step8. Return the result **IDS.Anomaly-found** (TRUE or FALSE).

• **Analysis of the complexity for the simple distributed algorithm:**

The total computation time is given by

$$\text{Sim-dist}(n) = T_{p1}(n) + (1/C_k) T_{p2}(n) + T_{\text{comm}} \quad (\text{EQ-5.2})$$

where, $T_{p1}(n)$ = running time of Phase-1 outlier detection

$T_{p2}(n)$ = running time of Phase-2 anomaly identification

C_k = constant indicating that the Phase-2 computing are partitioned among distributed hosts.

T_{comm} = communication time between the two kinds of processes, usually can be ignored.

Basically, the computation time for Phase-1 is almost the same as that of the sequential one. Hence, we find that the total computation time is decided by Phase-2. Since the computation is partitioned among different hosts, the total computation time can be improved by a constant factor $(1/C_k)$ over the sequential one.

4.4 Enhanced Distributed Two-Phase Approach With C-IDS

In this subsection, an enhanced distributed Two-phase DNS anomaly detection algorithm with a C-IDS is proposed. For operation details, please refer to section 3. The algorithm is as follows:

• **An enhanced distributed two-phase DNS anomaly detection algorithm**

Step1. Execute the Phase-1 DNS outlier detection algorithm. If the returned result **IDS.Call_Phase2** = FALSE, repeat Step1.

Step2. For each DNS anomaly type, execute Phase-2 analysis (**IDS.Algo-mode** = Enhanced-distributed, **IDS.Anomaly-type** = 1, 2, 3, 4).

Step3. Join the results of each type.

Step4. Go back to Step1.

• **Analysis of the complexity for the enhanced distributed algorithm:**

The total computation time is computed from

$$\text{Enh-dist}(n) = T_{p1}(n) + p * [(1/C_k) T_{p2}(n) + T_{\text{comm}}] \quad (\text{EQ-5.3})$$

where, p gives the probability that the newly identified DNS entry has been reported to C-IDS ($0 < p < 1$),

$T_{p1}(n)$ = Running time of Phase-1 outlier detection,

$T_{p2}(n)$ = Running time of Phase-2 anomaly identification,
 T_{comm} = Communication time between the two kinds of processes, usually could be ignored.

The introduction of the C-IDS in the enhanced distributed algorithm might improve the entire phase-2 computation by probability p (i.e., with a configuration consisting of different DNS servers and IDS groups, which is not uncommon) over the simple distributed one. Since the lookup and updating of the related DNS entries will be conducted a limited numbers of times, it can be ignored while determining the complexity.

5. EXPERIMENTAL RESULTS AND ANALYSIS

To evaluate the performance, we have implemented the algorithms and applied them to a number of examples. The results are summarized in Table 3 below. On average, there will be typically about 2500 to 3000 permitted incoming DNS query packets (i.e. these metrics might vary at different time of the day). While there exist some outstanding DNS anomalies, a collection of 10000 packets (i.e. roughly double of the normal case) in a 2-minute time window is common. Table 3 shows the experimental results from the three different configurations.

Table 3. Comparisons of the different algorithms.

Mode	Computation time	Notes
Sequential	716 (t1 = 125, Ptr = 30, A = 561)	$T1 = T(p1) + T(p2)$
Simple distributed	686 (t1 = 125, Ptr = 30, A = 561)	$T2 = T(p1) + T(p2, \text{dominated})$
Enhanced distributed	425 (t1 = 125, p = 0.5, Ptr = 30, A = 561)	$T3 = T(p1) + p * T(p2, \text{dominated})$

Assume with a probability $p = 0.5$, not uncommon in a configuration of 3 different DNS servers and IDS groups, the computation time of the previous example gets shortened to 425 seconds ($125 + 0.5 * 561$ seconds). In this case, the enhanced distributed algorithm might have a speed of 1.684 ($716/425$) over the sequential algorithm. In a real distributed environment, the probability p could be higher. Also, when there are outliers, the traffic distribution will show concentration on the outliers, and the computation time of these parts will be much more shortened.

6. CONCLUDING REMARKS AND FUTURE WORK

In this paper, we have proposed a distributed Two-phase DNS anomaly detection model for identifying DoS-like or DDoS-like DNS anomalies to the DNS system. Three sets of algorithms, corresponding to different configurations, are proposed. The experimental result shows a speed up about 1.684 on the test example when running on an enhanced distributed architecture with C-IDS compared to the sequential one without C-IDS. Higher speedup might be attained because DNS anomalies make the traffic distribution more concentrated on the outliers and the computation will usually converge much more quickly. This will be the goal of future research.

ACKNOWLEDGEMENT

This work was supported in part by the MOE Program of Excellence Research under Grant 89-E-FA-04-1-4.

REFERENCES

1. P. Mockapetris, "Domain names-concepts and facilities," *RFCs 1034*, 1987.
2. P. Mockapetris, "Domain names-implementation and specification," *RFC 1035*, 1987.
3. T. Lunt, "A survey of intrusion detection techniques," *Computer and Security*, Vol. 12, 1993, pp. 405-418.
4. Kessler and Gary C., "Defenses against distributed denial of service attacks," 2000; URL: <http://www.sans.org/infosecFAQ/threats/DDoS.htm>.
5. J. L. Koh, "Recent developments and emerging defenses to D/DoS: The microsoft attacks and distributed network security," 2001; URL: <http://www.sans.org/infosecFAQ/DNS/developments.htm>.
6. Hanley, Sinead, "DNS overview with a discussion of DNS spoofing," 2000; URL: <http://www.sans.org/infosecFAQ/DNS/DNS.htm>.
7. N. Brownlee, K. C. Claffy, and E. Nemeth, "DNS Root/Gtld performance measurement," *USENIX Lisa Conference*, 2001.
8. D. R. Morrison, "PATRICIA-practical algorithm to retrieve information coded in alphanumeric," *Journal of the ACM*, Vol. 15, 1968, pp. 514-534.
9. K. Cho, R. Kaizaki, and A. Kato, "Aguri, an aggregation-based traffic profiler," *2nd International Workshop on Quality of Future Internet Services*, 2001, pp. 24-26.
10. BIND (Berkeley Internet Domain); URL:<http://www.isc.org>.
11. S. McCanne, C. Leres, and V. Jacobson, "Tcpcdump," Lawrence Berkeley National Laboratory, Network Research Group; URL:<http://www.tcpdump.org>.

Chang-Sheng Chen (陳昌盛) was born in Hsinchu, Taiwan on June 17, 1964. He graduated with a B.S. degree from the Department of Electrical Engineering, National Taiwan University, Taiwan in 1986. He received the M.S. degree from the Department of Computer and Information Science, National Chiao Tung University, Taiwan in 1993. Currently, he is a Ph.D. student at National Chiao Tung University, Taiwan. His current research interests include Internet-based applications, knowledge engineering, expert systems, and network intrusion detection, etc.

Shian-Shyong Tseng (曾憲雄) received his Ph.D. degree in Computer Engineering from the National Chiao Tung University in 1984. Since August 1983, he has been on the faculty of the Department of Computer and Information Science at National Chiao Tung University, and is currently a Professor there. From 1988 to 1992, he was the Director of the Computer Center National Chiao Tung University. From 1991 to 1992 and 1996 to 1998, he acted as the Chairman of Department of Computer and Information

Science. From 1992 to 1996, he was the Director of the Computer Center at Ministry of Education and the Chairman of Taiwan Academic Network (TANet) management committee. In December 1999, he founded Taiwan Network Information Center (TWNIC) and is now the Chairman of the board of directors of TWNIC. His current research interests include parallel processing, expert systems, computer algorithm and Internet-based applications.

Chien-Liang Liu (劉建良) was born in Tainan, Taiwan, on February 22, 1976. He received his B.S. and M.S. degrees from Department of Computer and Information Science at National Chiao Tung University in 1998 and 2000 respectively. Currently, he is a Ph.D. student at National Chiao Tung University, Taiwan. His current research interests include Internet-based applications, knowledge engineering, expert systems, and data mining etc.