# A P2P BOTNET VIRUS DETECTION SYSTEM BASED ON DATA-MINING ALGORITHMS

Wernhuar Tarng, Cheng-Kang Chou and Kuo-Liang Ou

Graduate Institute of Computer Science,
National Hsinchu University of Education, Taiwan
wtarng@nhcue.edu.tw, coolinstar@gmail.com, klou@nhcue.edu.tw

## *ABSTRACT*

*A P2P botnet virus detection system based on data-mining algorithms is proposed in this study to detect the infected computers quickly using Bayes Classifier and Neural Network (NN) Classifier. The system can detect P2P botnet viruses in the early stage of infection and report to network managers to avoid further infection. The system adopts real-time flow identification techniques to detect traffic flows produced by P2P application programs and botnet viruses by comparing with the known flow patterns in the database. After trained by adjusting the system parameters using test samples, the experimental results show that the accuracy of Bayes Classifier is 95.78% and that of NN Classifier is 98.71% in detecting P2P botnet viruses and suspected flows to achieve the goal of infection control in a short time.*

## *KEYWORDS*

*Data Mining, Bayes Classifier, Neural Network, P2P Botnet, Virus Detection Systems*

## 1. INTRODUCTION

As the fast development of information technology, the population of using the Internet increases rapidly. Due to the prevalence of social networks, a huge amount of data are transmitted via networks in every second, causing the potential risks of disclosing personal information. One can see on television or from newspapers about the news of hackers stealing confidential data for illegal usage, and they may use a variety of methods such as Distributed Denial of Service (DDoS), Spam and Trojan. These methods require the cooperation of many computers, so hackers often spread out malicious software to achieve the goal of attacks. Therefore, it is important to enhance network security while developing and applying new information technologies to prevent from illegal access to confidential information.

A botnet is a collection of software agents, or robots, that run autonomously and automatically [1]. The term is most commonly associated with IRC botnets and more recently malicious software, but it can also refer to a computer network using distributed computation software. Botnets are usually named after their malicious software, such as Peacomm and Waledac. Basically, the composition of a botnet includes: the server programs used to control the infected computers, the client programs installed on the infected computers waiting for the control instructions, and the malicious software to infect normal computers as zombie computers to steal confidential data. The above programs often use a unique encryption system to communicate with each other to prevent from detection.

A large amount of infected computers may own considerable computation power and network resources. According to the statistics of SRI's technical report [2], Storm Worm was first discovered in 2007 and it spread quickly within 8 months to form a large botnet by infecting 250,000 to 1,000,000 computers. Hackers used the vast amount of computers to conduct malicious activities such as: Spam, DDoS as well as stealing personal data and accounts. Besides, the botnet viruses can update automatically and search for the target near the infected computers to expand the controlled range.

As the popularization of broadband networks, the traditional client-server architecture can not afford the heavy traffics on networks, and the requirement of real-time access even degrades its service quality. Hence, the way of accessing shared data on the Internet has changed gradually from the client-server architecture into P2P architecture. Currently, there are many application programs using the P2P technology for information sharing, e.g., eDonkey, Foxy, BitTorrent and GoGoBox.

The data-transmission power of P2P architecture usually depends on its number of connections; a larger connection number means a larger network space and shared bandwidth. The ability of P2P networks in bearing workload and its high transmission speed facilitates network sharing, but it also causes serious network security problems. To provide a number of users with network connections and shared resources, the P2P architecture usually adopts open-source software, and some extensive and fault-tolerant programs can easily be revised as virus programs for extending the coverage of botnets.

The nodes in a P2P network are usually connected through an *ad hoc* network [3], and the main idea is to form a logical network through the existing physical network, rather than reconstructing a new physical network. No matter what kind of logical network structure is adopted, the clients still have to transfer data through the physical layer. For simplicity, the installation of P2P software is usually done by clicking on a button. When the users are not familiar with its setting, their personal information can easily be exposed to others on the Internet. Besides, their computers may be infected by botnet viruses if they don't have the sense of network security or there is no anti-virus software installed.

In a P2P botnet, any zombie computer can be a client or a server, and it connects to the botnet according to the peer list to from a reciprocal relationship within the network topology. Therefore, a P2P botnet doesn't need any particular server to download programs or receive instructions, and the hackers can launch attacks from any computers in the P2P botnet. Consequently, the detection and prevention of P2P botnets are more difficult and challenging. To guarantee network security, network managers usually set up an intrusion detection system to detect and isolate viruses for defensive purpose.

The concept of intrusion detection systems was first proposed by Anderson in 1980 [4]. He replaced the checking mechanism by providing network security personnel with the information of internal risks and threats by applying statistic methods to analyze user behaviour and flow characteristics to find out illegal access to system resources. Denning [5] defined the role of an intrusion detection expert system (IDES) as monitoring the events happening on computers or network systems, analyzing the features of matters related to security issues and reporting problems as they occur.

Lu et al. [6] considered that in the future botnets will be attached to existing network applications (e.g., IRC, HTTP and P2P) as well as some unknown applications for making attacks, so they suggested using the characteristics and behaviour of traffic flows to find out what kind of applications are attached and then identify the botnets through the classification of traffic flows based on a decision-tree model. Their study was focused on IRC botnets only and they didn't address the issue of P2P and HTTP botnets. In their approach, the characteristics of traffic flows were determined based on the payload, or ASCII (0-255) distribution. In order to reduce the complexity of identification process, string comparison was used to identify the packets of some recognized applications in advance. However, their approach could increase the identification time and thus affect the overall efficiency.

In detecting the intrusion of botnet viruses, Villamarín-Salomón and Brustoloni [7] found that some botnet viruses leave the query records with NxDomain, meaning that the Domain doesn't exist, when accessing DNS servers. Therefore, Mizoguchi et al. [8] detected botnet viruses based on this discovery, but it was not applicable to encrypted P2P botnets. As botnets change from IRC to P2P architecture, or even merge into hybrid architecture, the detection of botnets is even more difficult. Currently, most studies in detecting botnets are based on their behaviour and the resulted flows for detection and identification.

Strayer et al. [9] found that botnet viruses have grouping and strongly related characteristics during their communication stage, so they filtered out normal flows and used the remaining suspected flows for clustering and identification. Liu [10] proposed an adaptive mechanism for detecting a variety of P2P botnets, but it can only be applied in the stage when the botnet is launching attacks. The mechanism can achieve the goal of detecting botnet viruses, but there are some more computers infected by the time when the viruses are identified since launching attacks is the last stage of virus infection. Because there is an incubation period during the infection of P2P botnet viruses, the resulted damage is reduced if we can develop an efficient virus detection system to find out the viruses in the early stage of infection. Therefore, this study utilizes P2P flow identification techniques to monitor and filter traffic flows, hoping to isolate the infected computers quickly when they connect to the botnet.

The objective of this study is to design a P2P botnet virus detection system based on two data-mining algorithms. The system adopts real-time flow identification techniques to detect the data flows produced by P2P application programs and botnet viruses by comparing with known flow patterns in the database. It can detect the botnet viruses and suspected flows quickly using Bayes Classifier and Neural Network (NN) Classifier in the early stage of infection to locate the IP's of infected computers and report to network managers to achieve the goal of infection control within a very short time.

## 2. RELATED METHOD

This study combines the techniques of P2P flow identification and data-mining algorithms to design a virus detection system and the objective is to determine if the network flows belong to P2P botnet viruses according to their behaviour and characteristics. There are two outcomes after classification, i.e., normal and abnormal flows. The former belong to some known P2P application programs and P2P botnet viruses, and the latter are from suspected P2P botnet viruses. The flow identification techniques adopted in this study and their theoretical bases are described in the following.

### 2.1. Characteristics of P2P Traffic Flows

P2P application programs typically use UDP protocols to establish connections during the communication stage and they will try to communicate with other computers in the peer list. However, not all P2P application programs use UDP protocols for connection. For example, GoGoBox uses TCP packets to initiate a reliable connection by three-way handshaking directly. TCP is a connection-oriented and reliable transmission protocol with a lower transmission speed.

When doing so, the computer will send out the packets with PSH=1 and ACK=1 to the P2P network for communication, so the data field of the packets with such strings can be retrieved for identification.

The behaviour of P2P botnets can be divided into the following four stages: (1) infection stage: inducing users to click on malicious links or open the attachments, (2) connection stage: the infected computer connecting to P2P botnets to receive commands or download programs, (3) download stage: proceeding with secondary infection or receiving commands, and (4) attack stage: starting attacks or spreading spam to the target hosts or specified computers. Since the packets of different P2P application programs and botnet viruses are different, the characteristics of their traffic flows are also different and thus can be used for identification. To satisfy practical requirements, this study analyzed the flow characteristics by some commonly used P2P application programs and the following three P2P botnet viruses:

- Trojan.Peacomm: The botnet virus spreads quickly in a short time to form a large botnet, and it uses Kademlia P2P network by the implementation of Distributed Hash Table (DHT). Its communication stage is similar to that of BitTorrent, where the connection with botnets is done by sending a large amount of small packets.
- W32.Waldac: The connection is done in the communication stage by sending out TCP packets with PSH=1 and ACK=1 to the P2P botnet.
- W32.Pilleuz: After infecting a computer, it will update itself within 5 to 6 minutes by downloading a new version of virus codes and start to spread through USB, real-time broadcasting and P2P networks. Its communication stage is very short, and it will open Port 6464 to provide HTTP connections to Symantec website in the second infection.

## 2.2. Data Mining

In this study, the Analyze Service in Microsoft SQL Server 2008 is used as the tool for data mining. It provides complete and powerful algorithms and thus can be used to build a project for dynamic solutions. As for data-mining algorithms, this study used Bayes Classifier and NN Classifier for the identification of P2P botnet viruses, and their theoretical bases are described in the following:

### (1) Bayes Classifier

Native Bayes is a classifier based on probability and statistics. In many applications, the data are collected gradually, so it is suitable for the applications with incremental data. The basic theory of Bayes Classifier is from Bayesian theorem and its formula is described below:

$$P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)},$$

where *X* represents unknown data and *C* stands for a known class. The meaning of this equation is that the probability of *X* data belonging to *C* class is equal to (the probability that *X* data appears in *C* class) × (the probability that *C* class appears) / (the probability that *X* data appears).

Bayesian network was proposed by Cooper and Herskovits in 1992 [11]. It is a search-and-score algorithm, which creates an independent Bayesian network by the combination of variable nodes. The largest Bayesian network can be found by calculating the structure score. The operation of Bayesian classification is done by the following procedure:

- There are no parent nodes for the initial status of each record.
- Increase parent nodes under the condition that the structure score can also increase.
- The procedure stops if there are no more parent nodes for adding.

The parent nodes mentioned above represent the splitting nodes of variables. A set of parent nodes $\pi_i$ belongs to the estimated value of variable *i*, and it can be expressed as $g(I,\pi_i)$ by the following equation:

$$g(I,\pi_i)=\prod_{j-1}^{qi}\frac{(ri-1)!}{(Nij+ri-1)!}\prod_{k-1}^{ri}Nijk!$$

The algorithm can achieve the optimal network structure by adding parent nodes gradually. During applications, it is better to find out the best splitting nodes for the variables and then create the best structure by adding parent nodes. There are two major stages for Bayesian classification: the first stage uses conditional probability to find out the dependence of different characteristics, i.e., to associate the characteristics with a certain virus; the second stage uses the known characteristics and the equation of conditional probability to identify botnet viruses. The procedure for detecting botnet viruses by Bayesian network is described below.

This study uses UDP packets, the total IP number and port number for outward connections as the related characteristics for predicting botnet viruses (Figure 1). The major part of Bayesian Classifier is to compute the probability that *X* appears in the known *C* class, i.e., P(*X*|*C*). The definition for the training equations in this study is described as following:

- P(UDP) is the probability that UDP packets appear in all packets.
- P(dIP) is the probability that a certain IP appears in all destination IP’s.
- P(dPort) is the probability that a certain Port address appears in all Port addresses.
- P(ratio) is the ratio of total number of Port addresses over the total number of outward connections.

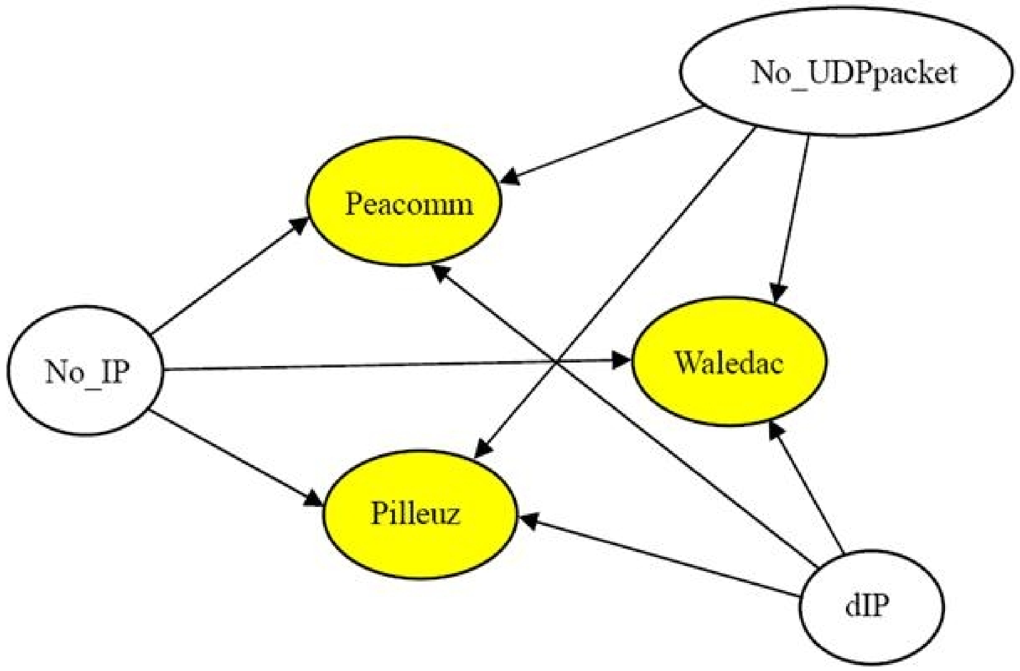

Figure 1. Bayesian dependency diagram based on flow characteristics

The training samples used in study include four commonly-used P2P application programs and three P2P botnet viruses, where W32.Pilleuz is treated as unknown virus. A total of 30,000 known and unknown traffic flows were generated for training the classifier to obtain the decision values according to the following conditional probability equations:

$V_m(UDP)=P_m(UDP)/P_0(UDP)$, $V_m(dIP)=P_m(ratio)/P_0(dIP)$ and $V_m(dPort)=P_m(ratio)/P_0(dPort)$,

where $P_m(UDP)$ is the probability that UDP packets appear under real-time monitoring. A virus is said to be detected if the following conditions are all satisfied:

$$P_m(ratio)>0.87,\ V_m(UDP)>0.99,\ V_m(dIP)>80 \text{ and } V_m(dPort)>80.$$

The decision values have a great influence on the identification accuracy, and some of the values can be fine-tuned by experienced system designers.

**(2) NN Classifier**

An artificial neural network (ANN) is a computing system, implemented by software or hardware, using a large number of interconnected artificial neurons to simulate the functional aspects of biological neural networks. In 1985, Mcclelland and Rumelhart [12] proposed a method for processing parallel-distributed information to investigate the fine structure of human cognition. The method is called Back-Propagation Neural Network (BPN), which includes input layers, hidden layers and output layers.

The operation of BPN can be divided into two stages, i.e., learning and back propagation. The former is a supervised learning process to obtain the expected output values by repeatedly input training samples to the network. The objective is to modify the parameters by learning processes for reducing the errors between the expected outputs and the inferred outputs. After training, the BPN can achieve an accurate result very close to the expected result.

The basic concept of BPN is to minimize the output error, and it is used in this study as the comparison and classification algorithm. It is required to select parameters before constructing a neural network. One can also use a decision-tree algorithm to develop the model and then uses the parameters determined by the decision tree as inputs. In this study, the UDP packets, total number of IP addresses and Port addresses for outward connections as well as the probability P(ratio) are defined as the characteristics for constructing NN Classifier (Figure 2).

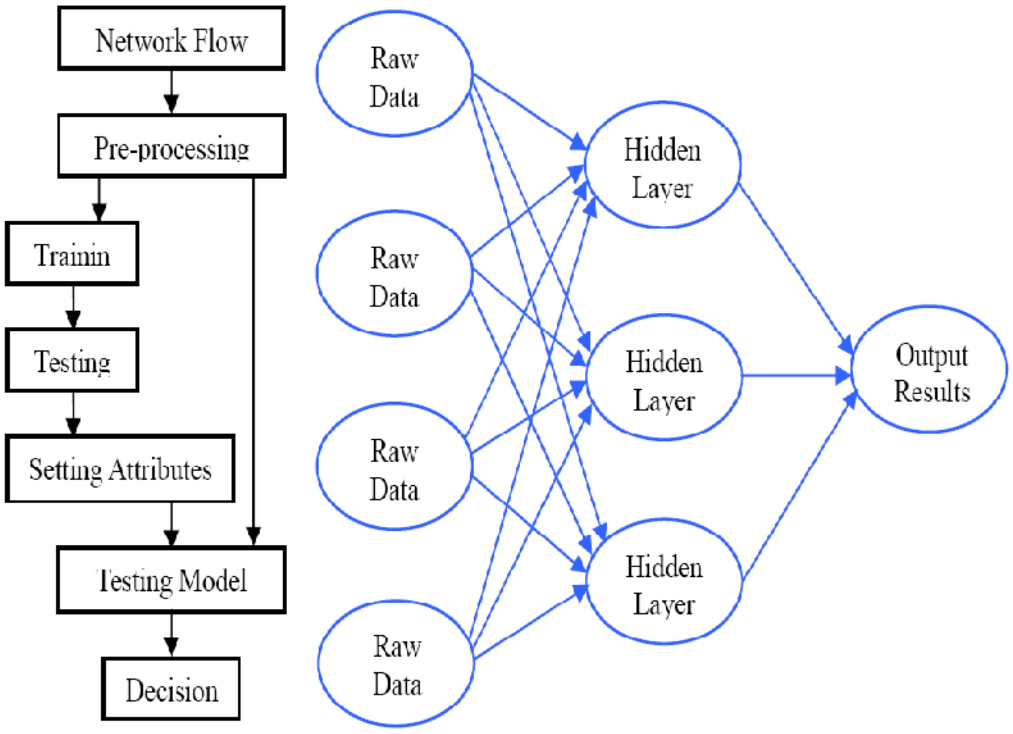

Figure 2. The architecture of NN Classifier

The hidden layers have a great influence on the learning of BPN. Usually, more hidden layers can achieve smaller errors, but it also takes more time for convergence during computation. When the hidden layers exceed a certain number, they are not useful in reducing the errors. There are two methods in determining the number of hidden layers. One is to use a lot of hidden layers for training at the beginning, and then reduce the number gradually. The other is to start with a few hidden layers for training, and then increase the number gradually. The unique goal for both methods is to minimize the errors.

The NN Classifier proposed in this study contains four input parameters and one output parameter (Table 1). To avoid over fitting by using too many hidden layers, the number of hidden layers is determined as 2 × (input layers + output layers). The output layer is used to determine if the flows belong to botnet viruses (1 meaning yes and 0 meaning no). In the training process, the training samples were fed into the NN Classifier until the convergence of root-mean-square error (RMSE) or the recursive training exceeds the preset number. Since the expected output is defined as 0 or 1, the thresh hold value for the output is set as 0.8, i.e., the output great than 0.8 is defined as abnormal flows. If the training result is not acceptable, the training process has to be done again until the acceptable solution is obtained.

Table 1. BPN input and output parameters

| No. | Parameter | Definition | Function |
|---|---|---|---|
| 1 | UDP | Number of UDP packets | Input |
| 2 | No_dIP | IP number for external connections | Input |
| 3 | No_dPort | Port number for external connections | Input |
| 4 | Ratio | Ratio of port number over IP number for external connections | Input |
| 5 | Y | Traffic flows by viruses | Output |

After the training of NN Classifier, the parameters have to be normalized to fall into the range of [0, 1] by the fallowing equation:

$$\text{normalized value} = (X - min) / (max - min),$$

where $X$ is the input parameter, *min* is the minimum and *max* is the maximum. When defining the parameter of UDP, it is classified as TCP or UDP by setting TCP as 1 and UDP as 0. After normalization, the input parameters No_dIP, No_dPort and Ratio all fall into the range of [0, 1].

## 3. System Design

The virus detection system proposed in this study adopts a two-stage mechanism for detecting P2P botnet viruses (Figure 3). The traffic flows obtained in the local area network (LAN) are directed into Monitor Module to filter out the known flows; the unknown flows are sent to Buffer Zone for classification by Bayes Classifier or NN Classifier. The traffic flows containing 30,000 packets were collected and divided into training samples and test samples for system training. The former are used for determining the decision values in Bayes Classifier and adjusting parameters in NN Classifier. According to the observation on infected computers and their traffic flows, the ratio of normal and abnormal flows is about 4:1, so there are 24,000 packets in normal flows and 6,000 packets in abnormal flows.

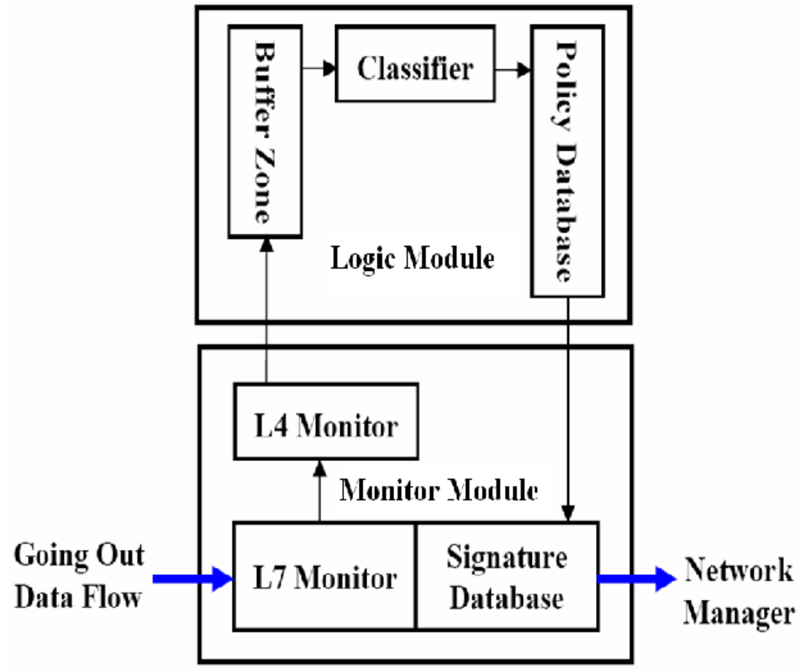

Figure 3. The architecture of P2P botnet virus detection system

When evaluating the performance of a virus detection system, the accuracy and responsiveness are the major issues. In this study, Bayes Classifier and NN Classifier were used for identifying botnet viruses and their performances were analyzed by simulation experiments. The results can be used as reference for selecting the more efficient classifier as well as the basis for further research on botnet virus detection systems. In the experiments, Four P2P application programs (BitTorrent, eDonkey, Foxy and GoGoBox) and three botnet viruses (W32.Waledac, Trojan.Peacomm and W32.Pilleuz) were used for testing and analysis.

The system proposed in this study can detect P2P botnet viruses after the infected computers enter the communication stage. Its architecture is divided into Monitor Module and Logic Module. The former analyzes the characteristics of outward flows and sends the suspected packets to the latter for identification. The functions in each module are described below.

## 3.1 Monitor Module

- L7 Monitor

L7 Monitor uses the well-known ports (0~1023) defined by the Internet Assigned Numbers Authority (IANA) to filter out non-P2P packets to speed up the identification process. The function can be used to identify and filter out the packets produced by non-P2P application programs to reduce the data amount in later processing. It ignores Port 80 and Port 443 because P2P application programs usually communicate through these ports. The filtered flows are then compared with Signature Database and no action is taken if the flows are safe. The known unsafe flows will be recorded and reported to the network manager, and the remaining flows are copied to L4 Monitor.

- L4 Monitor

L4 Monitor receives the packets from higher layers to identify if they are produced by P2P application programs. If the flows can not be identified after comparing with the characteristics in Signature Database, they are sent to Buffer Zone for temporary storage. The major function of L4 Monitor is to identify the software communicating with other computers using UDP packets. Since GoGoBox uses TCP packets for communication, it will also be identified as a P2P application program. L4 Monitor divides traffic flows into several groups according to their source ports, and tries to detect the flows in each group. If the packet sizes differ too much, the flows are sent to Buffer Zone for later processing.

- Signature Database

Signature Database is used to store the characteristics of P2P application programs and known P2P botnet viruses for the comparison with outward traffic flows. The flows are identified as attacks if they match the characteristics of a certain P2P botnet virus.

### 3.2 Logic Module

- Buffer Zone

  It is used to store the data from L4 Monitor temporarily.

- Classifier

When traffic flows are sent to Buffer Zone, the system uses Bayes Classifier or NN Classifier to identify if they are botnet viruses or not. The classifiers can filter out the unsafe flows and find out the infected computers or the computers producing the abnormal flows according to the related information of traffic flows. If the flows are from application programs, they are used as samples for training the decision tree. Otherwise, the flows are identified as unknown botnet viruses, and the system will report to network managers for isolating the infected computers to prevent from further infection.

- Policy Database

After ensuring the safety by using Bayes Classifier or NN Classifier, the results are compared with Policy Database. If the decision is unsafe, it is reported to network managers. If the decision is safe, no action will be taken.

## 4. Simulation Experiments

The following experiments are conducted to analyze if the proposed system can detect P2P botnet viruses effectively, and the results by Bayes Classifier and NN Classifier are compared based on their accuracy and reaction time. The experimental environment, procedure and results are described in the following.

### 4.1. Experimental Environment

This study adopted VMware to construct a virtual LAN for simulation experiments. The experimental equipment includes 10 computers, an Ethernet switch, a router and the proposed P2P botnet virus detection system (Figure 4). One or two P2P application programs (BitTorrent, Foxy, GoGoBox and eDunkey) were executed on each computer, and eight computers were infected intentionally with one to three botnet viruses (Trojan.Peacomm, W32.Waledac, and W32.Pulleuz). The experimental environment was monitored without affecting the other users by recording the execution time, infection status and traffic flows on each computer, hoping to simulate a real network environment. The status of each computer was restored by VMware Snapshot after the completion of each experiment.

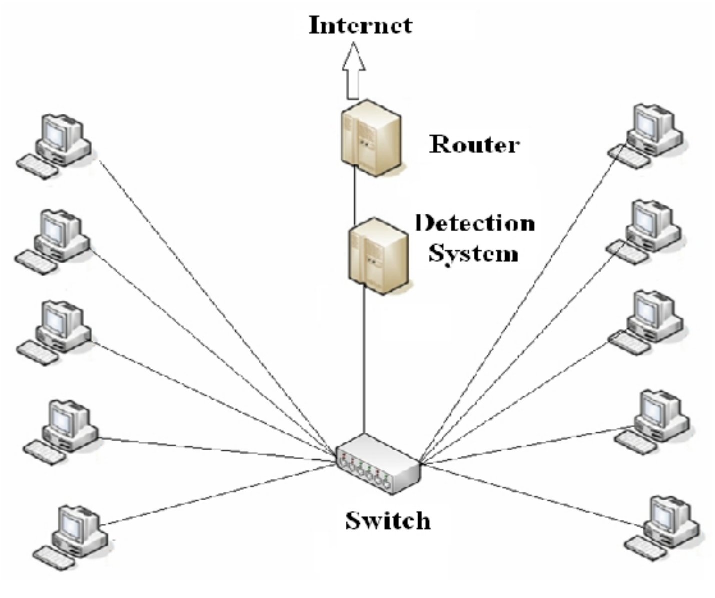

Figure 4. The network environment for simulation experiments

To analyze the performance in detecting unknown P2P botnet viruses by the proposed system, this study selected Trojan.Peacomm and W32.Waledac as known viruses and used their flows to train the decision-tree model. W32.Pilleuz was treated as the unknown virus and the simulation experiments mixed it with other P2P application programs, hoping that the classifiers can also detect it as well as the other known P2P botnet viruses. After the experimental environment was set up, the P2P botnet virus detection system started working for 10 minutes. In addition to the P2P application programs executed on each computer, some computers were infected by the botnet viruses as listed in Table 2.

Table 2. Programs executed on each computer in the simulation experiments

| Computer | Programs Executed | Packets in 10 minutes |
|---|---|---|
| 1 | P2P application programs BitTorrent and Foxy | 453600 |
| 2 | P2P application programs GoGoBox and eDunkey | 402000 |
| 3 | P2P application program BitTorrent<br>Infected by P2P botnet virus Trojan.Peacomm | 334535 |
| 4 | P2P application program Foxy<br>Infected by P2P botnet virus W32.Waledac | 179400 |
| 5 | P2P application program eDunkey<br>Infected by P2P botnet virus W32.Pulleuz | 377820 |
| 6 | P2P application program GoGoBox<br>Infected by P2P botnet virus Trojan.Peacomm and W32.Waledac | 195217 |
| 7 | P2P application program BitTorrent<br>Infected by P2P botnet virus W32.Waledac and W32.Pilleuz | 381580 |
| 8 | P2P application program Foxy<br>Infected by P2P botnet virus Trojan.Peacomm and W32.Pilleuz | 258911 |
| 9 | P2P application program eDunkey<br>Infected by P2P botnet virus Trojan.Peacomm, W32.Waledac and W32.Pulleuz | 441642 |
| 10 | P2P application program GoGoBox<br>Infected by P2P botnet virus Trojan.Peacomm, W32.Waledac and W32.Pulleuz | 222569 |

## 4.2. Experimental Procedure

The objective of this study is to develop a virus detection system to identify P2P botnet viruses quickly and accurately. The major part for identifying viruses is the classifier in Logic Module, where the flows are compared by their characteristics with those in Signature Database. The flows by known application programs can pass Monitor Module and only unsafe or unknown flows are directed into Logic Module. Therefore, the analysis is focused on the classification results. In this study, the confusion matrix proposed by Kohavi and Provost [13] was used to evaluate the classification results (Table 3).

Table 3. The confusion matrix to evaluate the classification results

| | Predicted Abnormal | Predicted Normal |
|---|---|---|
| **Actually Abnormal** | True Positive | False Negative |
| **Actually Normal** | False Positive | True Negative |

The parameters listed in Table 3 are defined in the following:

- True Positive(TP)

  The number of virus flows identified as virus flows
- True Negative(TN)

  The number of normal flows identified as normal flows
- False Positive(FP)

  The number of normal flows identified as virus flows
- False Negative(FN)

  The number of virus flows identified as normal flows

The accuracy of identification is defined as the ratio that the unsafe flows are identified as unsafe flows and the safe flows are identified as safe flows, and it can be computed by the following equation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

The simulation experiments adopted the training-and-testing method which divided the collected flows into two parts. 2/3 of the flows (training data) were used to construct the system model, and 1/3 of the flows were used as test data to ensure the accuracy of the system model. For the experiments using Bayes Classifier, the adjustment of decision values must rely on experiences and it may take a long time. The results of confusion matrix for training samples and test samples are shown in Table 4.

Table 4. Training and test samples and the results by Bayes Classifier

| **Samples** / **Flows** | **Training Sample** | | **Test Sample** | |
|---|---|---|---|---|
| | Known Flows | Abnormal Flows | Known Flows | Abnormal Flows |
| Known Flows | 304 | 7 | 152 | 3 |
| Abnormal Flows | 68 | 3621 | 35 | 1810 |

In the experiments using Bayes Classifier, this study simulated the conditions of single flows and multiple flows. According to the roles played by the computers listed in Table 2, the unknown virus W32.Pilleuz existed on five computers. During its communication stage, the virus detection system recorded the changes of UDP flows and directed them into Bayes Classifier for identification according to the selected characteristics. The accuracy of the test results is computed as 90%.

In the experiments using NN Classifier, this study trained over 50 times to find out the parameters for the best solution where the number of hidden layers=4 and threshold=0.8. Using these parameters to conduct experiments, the RMSE for the output was converged to 0.089 and the results of confusion matrix for training samples and test samples are shown in Table 5. The test samples were fed into the trained NN Classifier for the detection of the unknown flows and the accuracy of the test results is computed as 92%.

Table 5. Training and test samples and the results by Neural-Network Classifier

| **Samples** / **Flows** | **Training Sample** | | **Test Sample** | |
|---|---|---|---|---|
| | Known Flows | Abnormal Flows | Known Flows | Abnormal Flows |
| Known Flows | 238 | 6 | 114 | 3 |
| Abnormal Flows | 54 | 3702 | 26 | 1857 |

## 4.3. Experimental Results

The following experiments compared the accuracy of Bayes Classifier and NN Classifier in detecting P2P botnet viruses, which is defined as the ratio that P2P application programs and P2P botnet viruses are both identified correctly. The results in Table 6 show that the accuracy of NN Classifier (98.71%) is slightly higher than that of Bayes Classifier (95.78%).

Table 6. Comparison of accuracies between Bayes Classifier and NN Classifier

| **Computer** | **Bayes Classifier (%)** | **NN Classifier (%)** |
|---|---|---|
| 1 | 100.00 | 100.00 |
| 2 | 100.00 | 100.00 |
| 3 | 93.21 | 100.00 |
| 4 | 100.00 | 100.00 |
| 5 | 95.60 | 98.91 |
| 6 | 91.90 | 97.47 |
| 7 | 96.60 | 98.00 |
| 8 | 100.00 | 100.00 |
| 9 | 91.21 | 96.40 |
| 10 | 89.24 | 96.36 |
| **Average** | **95.78** | **98.71** |

In addition, this study tried to investigate the reaction time to P2P botnet viruses by the two classifiers. According to the results in Table 7, the time required to identify the known viruses by Bayes Classifier is 3 to 4 minutes, and is about 5 minutes for the unknown flows. When NN Classifier was used, the identification speed was higher and the time required for known viruses is 1 to 2 minutes, and is about 2.5 minutes for the unknown flows.

Table 7. The reaction time by Bayes Classifier and NN Classifier

| Computer | Program | Bayes Classifier | | NN Classifier | |
|---|---|---|---|---|---|
| | | Flow | Reaction Time | Flow | Reaction Time |
| 1 | BitTorrent | 299400 | - | 287400 | - |
| | Foxy | 161230 | - | 169910 | - |
| 2 | GoGoBox | 91218 | - | 90031 | - |
| | eDunkey | 310082 | - | 324500 | - |
| 3 | BitTorrent | 284817 | - | 249910 | - |
| | Trojan.Peacomm | 39718 | 187 | 40007 | 67 |
| 4 | Foxy | 159235 | - | 143317 | - |
| | W32.Waledac | 26121 | 202 | 21800 | 80 |
| 5 | eDunkey | 301000 | - | 254030 | - |
| | W32.Pulleuz | 66420 | 291 | 64473 | 140 |
| 6 | GoGoBox | 143270 | - | 94830 | - |
| | Trojan.Peacomm | 37860 | 200 | 41120 | 68 |
| | W32.Waledac | 24130 | 248 | 21968 | 89 |
| 7 | BitTorrent | 296753 | - | 276000 | - |
| | W32.Waledac | 25200 | 210 | 22273 | 83 |
| | W32.Pilleuz | 61247* | 288 | 63700 | 145 |
| 8 | Foxy | 161360 | - | 174303 | - |
| | Trojan.Peacomm | 38145 | 193 | 40210 | 67 |
| | W32.Pulleuz | 67395 | 304 | 64221 | 160 |
| 9 | eDunkey | 320900 | - | 276000 | - |
| | Trojan.Peacomm | 41373 | 220 | 42740 | 71 |
| | W32.Waledac | 24697 | 247 | 26820 | 93 |
| | W32.Pulleuz | 65820 | 433 | 63371 | 158 |
| 10 | GoGoBox | 80380 | - | 91160 | - |
| | Trojan.Peacomm | 40410 | 191 | 36147 | 67 |
| | W32.Waledac | 26800 | 248 | 25800 | 95 |
| | W32.Pulleuz | 64100 | 310 | 62813 | 166 |

According to the experimental results, both classifiers can achieve the accuracy of over 95% and the accuracy of NN Classifier is slightly higher than that of Bayes Classifier. For the reaction time, both classifiers can detect P2P botnet viruses within 5 minutes and the time required by NN Classifier is about half the time required by Bayes Classifier. The P2P botnet virus detection system was designed using SQL Server for virus detection and analysis. Some parameters of NN Classifier are hidden and therefore not easily adjusted, so Bayes Classifier is easier than NN Classifier in finding the optimal solution during the training stage.

## 5. CONCLUSIONS AND FUTURE STUDY

In this study, a P2P botnet virus detection system is developed based on two data-mining algorithms, i.e., Bayes Classifier and NN Classifier. When the system is installed on the network, it can monitor the traffic flows to detect the infected computers and identify the P2P botnet viruses in real time. Simulation experiments were conducted to measure the accuracy of the virus detection system. The results show that the system can identify known or unknown flows produced by P2P botnet viruses correctly in a short time to achieve the goal of infection control. After trained by adjusting the system parameters using test samples, the accuracy of Bayes Classifier is 95.78% and that of NN Classifier is 98.71%. Both classifiers can identify known and unknown P2P botnet viruses within 5 minutes. The system can also be implemented by hardware to speed up the identification in the future.

This study is focused on a LAN environment, so the major concern for the virus detection system is its accuracy and responsiveness. When considering the efficiency under high-traffic WAN environments, it is suggested to use a distributed virus detection system to shorten the reaction time. In order to detect the infected computers in a large WAN environment, the virus detection system has to be installed on the ISP site, and the flows in the LAN environments may be directed into the same IP if NAT technique is applied. In that case, the identification of infected computers is more complicated and this problem is left for the future study.

## Acknowledgement

The authors wish to thank for the financial support of the National Science Council of the Republic of China under the contract number NSC 101-2631-S-008-001.

## REFERENCES

[1] Schiller, C., Binkley, J. and Harley, D. (2007). Botnets: The killer web applications, Rockland, MA: Syngress Publishing, Feb. 2007.

[2] Porras, P., Saidi, H. and Yegneswaran, V. (2007). A multi-perspective analysis of the Storm Worm, Technical Report, Computer Science Laboratory, SRI International, Oct. 2007.

[3] Lee, S. T. (2008). Design and implementation of P2P traffic flows management system, Master Thesis, Department of Information Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan.

[4] Anderson, James P. (1980). Computer Security Threat Monitoring and Surveillance, James P. Anderson Co., Fort Washington, PA.

[5] Denning, Dorothy E. (1987). An Intrusion-Detection Model, IEEE Transaction on Software Engineering, 13(7), 222-232, 1987.

[6] Lu, W., Tavallaee, M., Rammidi, G. and Ghorbani, A. (2009). BotCop: an online botnet traffic classifier, 7th Annual Conference on Communication Networks and Services Research, Moncton, Canada, May 11-13, 2009.

[7] Villamarín-Salomón, Ricardo and Brustoloni, José Carlos (2009). Bayesian Bot Detection Based on DNS Traffic Similarity, 24th Annual Symposium on Applied Computing-Computer Security Track (SAC 2009), ACM, Honolulu, HI, Mar. 8-12, 2009.

[8] Mizoguchi, S., Kugisaki, Y., Kasahara, Y., Hori, Y., and Sakurai, K. (2010). Implementation and evaluation of bot detection scheme based on data transmission intervals, IEEE Workshop on Secure Network Protocols, Oct. 5, 2010.

[9] Strayer, W. T., Walsh, R., Livadas, C. and Lapsley, D. (2006). Detecting Botnets with Tight Command and Control, IEEE Conference on Local Computer Networks, Nov. 15-16, 2006.

[10] Liu, P. W. (2009). An Adaptive Defence Mechanism for P2P Botnet, Master Thesis, Department of Communication Engineering, Chun Yuan Christian University, Chung Li, Taiwan.

[11] Cooper, G. F. and Herskovits, E. (1992). A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning, 9, 309-347, 1992.

[12] McClelland, J. L. and Rumelhart, D. E. (1985). Distributed Memory and the Representation of General and Specific Information, Journal of Experimental Psychology: General, 114(2), 159-188, 1985.

[13] Kohavi, R. and Provost, F. (1998). Glossary of terms, Machine Learning, 30(2/3), 271-274, 1998.