SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2015; **8**:459–470
Published online 9 June 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.993

RESEARCH ARTICLE

Design and implementation of a malware detection system based on network behavior

L. Xue[1] and G. Sun[2]*

[1] College of Computer, Nanjing University of Posts & Telecommunications, Nanjing, China
[2] Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing, China

ABSTRACT

With the increasing of new malicious software attacks, the host-based malware detection methods cannot always detect the latest unknown malware. Intrusion detection system does not focus on malware detection, whereas the behavior-based detection methods still have some difficulties in being deployed in the network layer. This paper presents a malware detection method based on network behavior evidence chains. The proposed new method will detect the specific network behavior characteristics on three different stages as connection establishment, operating control, and connection maintenance. Then a final detection decision will be concluded according to the results detected in the different stages before. A system prototype is implemented to proof concept the proposed malware detection methods. Copyright © 2014 John Wiley & Sons, Ltd.

KEYWORDS
traffic detection; behavioral analysis; traffic classification; probability analysis

***Correspondence**
Guozi Sun, College of Computer, Nanjing University of Posts & Telecommunications, Nanjing, China.
E-mail: sun@njupt.edu.cn

1. OVERVIEW

The Internet communication technology brings great convenience to our society. At the same time, a variety of viruses and malicious codes increasingly spread on the network seriously, it brings growing threats to the network. Among the various forms of malicious software, the Trojan and spyware are most popular in internet. These malwares are always used or programmed by attackers to gain access to private computer systems or gather sensitive information. Trojan and spyware distinguish themselves from other forms of malware, by that, they have to establish a command and control channel to translate command and data. Most of the current malware detection technology runs on one machine and cannot monitor the network layer effectively. So such kind of malware detection is not beneficial for detecting the whole local area network (LAN). Some intrusion detection system (IDS)-based network malware detecting methods, such as Snort [1], is based on the communication ports and some other known characteristics. These methods do not identify or detect the communication behavior. Thus, these methods are sometimes ineffective with the malicious software that changes the ports dynamically and disguise themselves with protocols. They are also not effective for some unknown malicious software newly created.

In this paper, we present a novel malware detection technique with system implementation that overcomes some of the limitations of existing anti-malware approaches. The new technique aims at identifying the internal hosts infected by malwares through inspecting the traffic cross of the border of the whole network. We exploit this new technique with the observation that the malwares share certain characteristics in their network behaviors that are distinct from those of normal software. The proposed technique is based on abstract characteristics of the network behavior of the popular malware and Bayes classification algorithm [2].

2. RELATED WORK

The current malware detection method can be classified into host-based detection and intrusion detection as well as network behavior-based detection methods.

2.1. Host-based detection

Host-based malware detection can be realized via three methods as follows: (i) signature detection [3,4]. The malicious software can be identified by comparing the static signature of the suspicious files with the signature of

known malware in the feature library. The false positive rate of this method is low, but it is difficult to collect the signature and it is slow to update. So this detection method cannot identify the unknown malicious software. (ii) integrity detection [5,6], it can detect malicious software effectively when they inject the host and modify the file system. It is generally used to prevent the malware from injecting the host and cannot detect the injected malware. Besides, the false positive rate and the false negative rate of this method are high. (iii) Heuristic detection [7–9]. By simulating the suspicious files, monitoring the system files (including the system regedit, documents, service, etc.), system processes and application programming interfaces, this technique can detect the suspicious behavior. It can be divided into static detection and dynamic detection, in which the difference is whether to run the detected documents to check malicious operation or not.

In summary, host-based detection technique cannot ensure finding all the Trojans, and it is hard to deploy on every computer of the whole network. If there are any computers that are not protected by this technique, it is possible that there may be malicious software spreading to the whole network from this computer.

2.2. Intrusion detection

At present, IDS and intrusion prevention system can enhance the malware detection of the network by adding special rules. But because the IDS is not specially designed for detecting the communication behavior of malicious software, the detection rate cannot be ensured. For example, Snort is configured with over 2500 detecting rules, and some other detection models in the network layer also adopt the similar methods.

The IDSs mainly detect the attacks on the internet infrastructure and the proliferation of unsolicited spam e-mails. It is difficult for them to detect hosts infected with malware such as Trojan horse, because generally, these kind of communications neither consume significant bandwidth nor involve a large number of targets. Analysis can be further complicated if a malware encrypts the network traffic, using rebound port [10] and communicating over peer-to-peer (P2P) protocols to blending with P2P file-sharing traffic [11].

Besides, a majority of antivirus firms and research organizations use honeypots to capture mutants of known and unknown malware. These products are based on the acquired data and the binary samples to produce patches for defense against malware threats. But their main aim is to detect and gather binary samples of mutants of existing malware and viruses [12] instead of malware such as Trojan horse.

2.3. Network behavior-based detection

Recent studies have proposed methods for detecting new or known malware using machine learning techniques. In [13], Moskovitch *et al.* proposed to detect the unknown computer worms based on behavioral classification of the host. Their detection accuracy exceeded 90%, and for specific unknown worms accuracy reached above 99%, while maintaining a low level of false positive rate. In [14], Moskovitch *et al.* have used active learning to detect malcode. Though they would like to expand their approach to other malcodes, such as viruses, spyware and adware, they just focused on worms. Bianchi *et al.* have designed Blacksheep, a distributed system for detecting a rootkit infestation among groups of similar machines [15]. It can detect the rootkit running in Windows 7 and Windows XP kernel efficiently and effectively. Xu *et al.* have developed a prototype system, PeerPress, combining both the robustness of host-based approaches and the efficiency of network-based approaches to detect P2P malware [16], but they only focus on P2P malware and it is unavailable when the traffic is encrypted.

Despite of being existed for a short time, the network behavior-based detection methods have demonstrated very effective in identifying malwares, especially the unknown malwares.

2.4. Our work

In this paper, we propose a new malware detection model based on network communication behavior. We name our new model DTrojan. DTrojan can effectively detect the communication behavior of malicious software on the stages of connection establishment in communications, operating control, and connection maintenance of communications using Bayes classification algorithm [2]. The DTrojan model has three stages: (i) network behavior of the collected malware is monitored in honeypot environment and the hosts with similar behaviors are grouped together; (ii) based on a corpus of known malware, a malware behavior classifier is trained using learning techniques; and (iii) the malware behavior classifier is used to detect and identify the behavior of malware in real network.

The proposed DTrojan model have several advantages: (i) network administrator can use DTrojan to detect and protect all the hosts in the network; (ii) DTrojan will not only identify known malware but also find new and unknown malware effectively; (iii) DTrojan will have both low false negative rate and false positive rate; and (iv) DTrojan can be expanded to detect other malware such as worms, viruses, and adwares besides spyware.

We have also implemented the DTrojan model into a prototype testbed named DTrojanDemo. The prototype is a lightweight system specifically used for detecting network malware, which is easy to configure on the LAN gateway. It collects and detects the network data in and out of the LAN and shows the detection result to the network administrator in time. As demonstrated from our experimental results, this prototype can find malicious software such as Trojan timely and effectively.

DOI: 10.1002/sec

The rest of this paper is organized as follows. Section 3 proposes the design of behavior-based malware detecting DTrojan model. Section 4 shows the performance result of the model DTrojan. Section 5 explains the detailed system prototype DTrojanDemo with the proposed algorithms implemented and with the experimental validations. Section 6 gives the conclusion of this work.

3. DTROJAN DESIGN

The major purpose of malware is to acquire the information from the infected host according to the commands from the attacker. As a result, the Trojan or spyware has to establish a command and control channel [17], which is a kind of persistent and end-to-end connection to receive command from the attacker and sends data to the attacker and its typical client–server communication mode. In general, the communication process of a malware can be divided into three stages: connection establishment, operating control, and connection maintenance. In all stages, the malwares connection would have some unique features. So we can collect these features and use them to train the malware behavior classifier to detect the malware.

The detected data flow can be divided into two classes. One is the tentative connection data flow that is not successful to connect because the network or remote host is not online. This kind of data flow happens when source host continuously sends synchronization character (SYN) packets to remote host, but transfer control protocol (TCP) connection is not successfully established between them. These traffic packets have the same source Internet Protocol (IP) address, destination IP address, and destination port. Because of connection failure, source IP address would try to establish the connection with destination IP address from different ports, thus the source ports are different. We can use source IP address, destination IP address, destination ports, and detecting time as the features of a connection.

The other one is the communication data flow that is generated when TCP three handshake session is finished and connection is successfully established. Such kind of data flow is actually a TCP session. We can use the five tuples (namely source IP address, source port, destination IP address, destination port, and detecting time) to identify this kind of data flow [18].

When the control side is not online, the malware would periodically try to establish a connection with the monitoring port of control side. Therefore, the tentative connection data flow is generated. When in operating control and connection maintenance, the data flow from malware is the communication data flow, with large differences from the normal flow in packet length and traffic data.

We illustrate the test environment of the implemented prototype DTrojanDemo in Figure 1. The details of the prototype design will be presented in Section 5. In order to train the model and evaluate the features used by DTrojan, we use seven different samples of malware run on different virtual machines (honeypot) and send operating commands from the control host. Besides, there are also two hosts without any sample of malware. Besides, all the five clients are used to do the daily work normally. The data capture

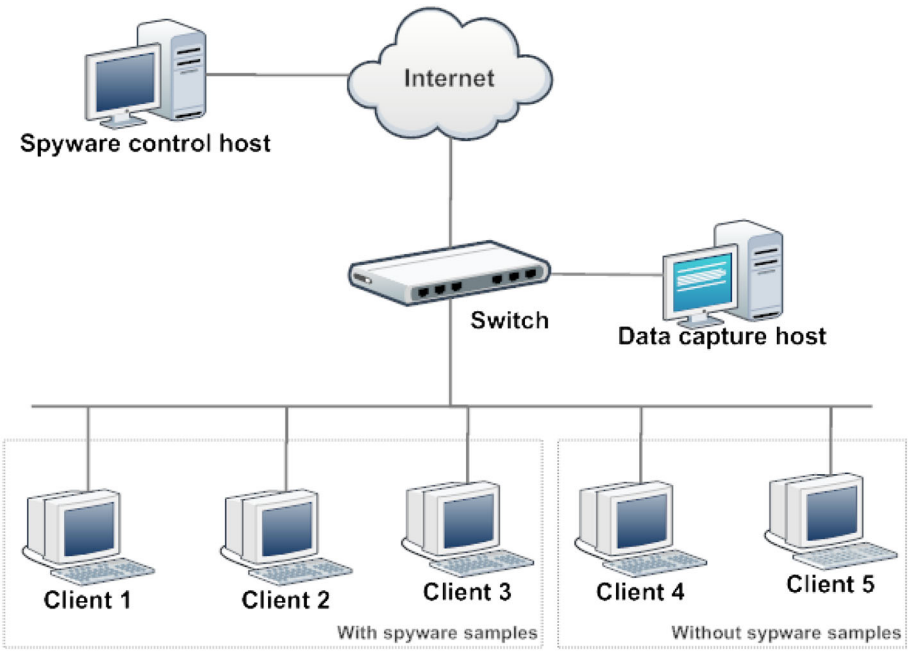

Figure 1. Test environment for capturing training data using DTrojan.

host connected to the mirror port of the switch is used to capture the network data of the LAN. Then we use the data capture host to collect network data from the mirror port for 1 day (24 h) as the training data. Because all malware samples use configured port to communicate with the malware control host, we can distinguish the sample malware communication data from the rest of the LAN data.

In the rest of this section, we use the training data to explain the details of the proposed DTrojan malware detection method.

3.1. Connection establishment

At this stage, most of the malicious software adopts ports back-bouncing [19] technology for pushing through the limit of firewall. So it is unlikely to detect the malware connection from the view of connection establishment. However, there are still many unchangeable features due to the need of transmitting data.

When the control side is offline or connection establishment fails, the malicious software would continuously sending SYN packets to the control port. The amount of sending packets and the time interval are similar to those regular connections, but most normal programs will give up connection after several trials. Figure 2 shows a measurement of the packet-sending time sequences that the controlled side tries to connect outside when pcshare [19] control side is offline.

In Figure 2, when sending data packets to the control side, the malware sends a series data packets C (several SYN packets) every time. In this paper, the packets with time interval smaller than threshold value t_{th} are regarded as a packet cluster. Though the control side has the fixed IP and listening port, if the tentative connection fails, the malware will change the source port and try again. Therefore, we can define that, in the detecting period T, the largest time interval between two adjacent packets, which are generated before the connection, is established as t_{max}. We have analyzed the Trojan horse samples that have this feature and find that the time intervals between adjacent packets belonging to same cluster are seldom larger than $\frac{1}{2}t_{max}$, as a result, we define that the relationship between t_{th} and t_{max} is defined as Equation 1.

$$t_{th} = \frac{1}{2}t_{max} \tag{1}$$

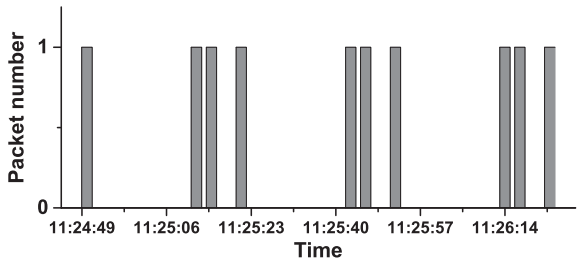

Figure 2. Pcshare tentatively connection packet sequence.

If the threshold value of packet cluster time interval t_{th} is ensured, the whole tentative connection can be regarded as a set of tentative connection packet clusters and represented by Equation 2, where n refers to the number of packet clusters in tentative connection data flow, C_n means the n^{th} packet cluster in SP, where n refers to the number of packet clusters in tentative connection data flow, and C_n means the n^{th} packet cluster in SP.

$$SP = \{C_1, C_2, \ldots, C_n\} \tag{2}$$

ΔTC represents the period between the adjacent packets and is computed by Equation 3.

$$\Delta TC_i = TFPC_{i+1} - TFPC_i \tag{3}$$

In Equation 3, $TPFC_i$ refers to the time stamp of the first packet cluster in C_i, with unit in seconds.

We consider that only when the number of packet clusters in set SP is large and the duration is long, then it is necessary to detect the tentative connection features. When the time interval is too short or the number of packet clusters is small, it is possibly a normal program other than malware tentative connection packets, and it connects the packets for once or more.

Therefore, only when $n > 10$ and the duration is larger than 10 min, we consider that this data flow has the detecting condition for tentative connection. We set the threshold value of the variance $SD_{\Delta tc}$ of $\Delta TC_1, \Delta TC_2, \ldots, \Delta TC_{n-1}$'s, TH_{sdt} as 1.0. Only when $SD_{\Delta tc} > TH_{sdt}$, we consider that this data flow has the tentative connection feature, and the probability is defined as $TrojanFProb_{tc}$, which will be further discussed in Section 3.4.1.

3.2. Operating control

After successful communication establishment, the next stage is the operating control. It means that the attacker will send control commands, such as file search, file download, screen capture, and sound monitor, to the malicious software. Then malware would accomplish the related operation according to the control commands and sends the result and data to the control side. Table I shows some upstream and downstream correlation data in the operating control stage measured from the training data. In general, the data flow in this stage has larger average upload stream and larger average upload packet length.

3.2.1. The analysis of upstream and downstream data flow.

For the communication process between malware and control side, the control side sends control commands to malware, and malware returns large amount of sensitive data to the control side [20]. Therefore, the downstream traffic is far less than the upstream traffic in the communication data flow of malware, whereas in the normal data,

Table I. The comparison of malware upstream and downstream data.

Trojan horse	Up packets	Down packets	Down data (KB)	Up data (KB)	Average uppkt_len (bytes)	Average downpkt_len (bytes)
Pchare	3925	2355	3677	142	936	60
PoisonIvy	7136	4744	4935	383	691	80
Huigezi	5212	3542	4234	238	812	67
Icyriver	3222	2198	3432	223	1065	101
Magicremote	5367	3432	4521	306	842	89
Netsys	3123	2523	2931	143	935	76

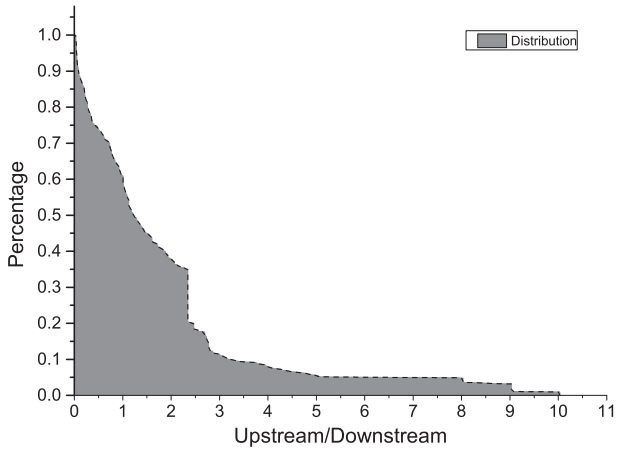

Figure 3. The accumulative probability distribution of upstream over downstream.

such as web browsing and file downloading, the downstream traffic is generally much more than the upstream traffic. In Table I, where we have the statistics of six malware upstream and downstream data flows on the operating control stage, it is easy to obtain the conclusion that the upstream data amount is obviously larger than the downstream one. The proportion between the upstream data amount and the downstream data amount (*Upstream_data*/*Downstream_data*) is 25.9 12.9 17.7 15.4 14.8 20.5 , respectively for the six types of Trojan horse shown in Table I.

At the same time, we captured 2206 normal session streams, 857 of which are upstream and downstream bidirectional data. We show the accumulative probability distribution of these 857 session streams in Figure 3. There are only eight sessions in which the ratio of upstream traffic against downstream traffic is larger than 10, occupying only 1% of the total sessions.

In the following, we set TH_{udd}, the ratio threshold value of upstream data against downstream data in operating control data flow, as 9.0. When the ratio of the upstream data, $Upstream_{data}$, against downstream data, $Downstream_{datais}$, is larger than TH_{udd}, this data flow has the flow inversion feature of Trojan data flow. The abnormal data flow probability of such feature is defined as $TrojanFProb_{udd}$, which will be specified in Section 3.4.1.

3.2.2. The size detection for upstream and downstream data packets.

The major roles of the control side of a malware play are as follows: sending control commands and sending response packets after receiving the data from the controlled side. Therefore, the packets that control the side send are always small, whereas the packets that control the side send are large-sized data packets. In Table I, it is easy to obtain the conclusion that in the operating and control time, the average packet length of upstream data from almost all the malwares is more than 600 bytes, and the average packet length of downstream data is less than 110 bytes. Therefore, we set the threshold values of the aforementioned two lengths TH_{upl} and TH_{dpl} to 600 and 110 bytes, respectively. For a certain data flow, if its average upstream packet length is bigger than TH_{upl} and the average downstream packet length is smaller than TH_{dpl}, this data flow has the length feature of malware communication data packets, and the abnormal data flow probability is defined as $TrojanFProb_{pl}$, which will be specified in Section 3.4.1.

3.2.3. Interactive command detection.

After receiving the commands from the control side, besides sending data to the control side, the controlled side of malware may operate on all the hosts for shell

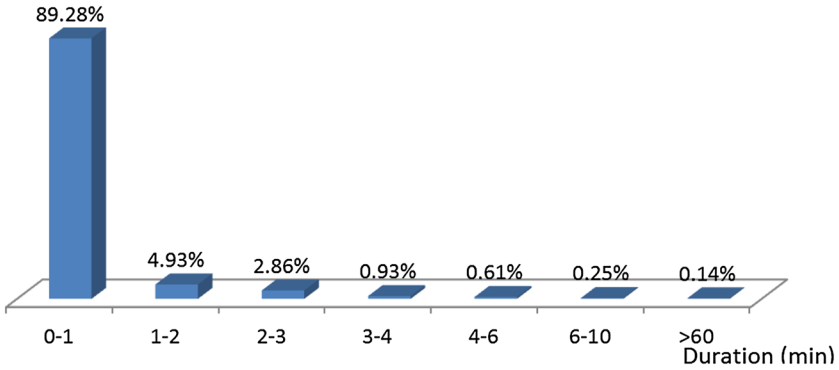

Figure 4. The duration distribution of normal connections.

commands. In interactive sessions of shell commands, the time interval of continuous small packets (in which length is less than 20 bytes) is between 10 ms and 2 s. When the time interval proportions are more than the specific threshold value TH_{ic} between 10 ms and 2 s, this data flow is considered to have the interactive command characteristic of a malware, and the abnormal dataflow probability is defined as $TrojanFProb_{ic}$. According to reference [21], when the detection result is good, TH_{ic} is set to 0.2.

3.2.4. Connecting time detection.

Because most of the normal data flows we used in this paper are generated by artificial operations in our lab and the generated time is during the working hours, a data flow generated outside the working hours has high probability to be generated by malware. Once connected, the malware would not disconnect it except that the control side is offline or the hosts are shut down. Thus, the connection time is always very long. We find out the normal connections from the training data and the distribution of these duration shown in Figure 4. In the duration distribution of 2279 normal connections, the ones whose continuous time over 60 min only occupy 0.14%; the ones over 5 min occupy about 0.5%. As almost all malware connections last more than 5 min, we set the threshold value of connection time as 5 min. When the connection duration of a specific data flow is over 5 min, this data flow would be considered to have the continuous time characteristic of a malware, and the abnormal data flow probability is defined as $TrojanFProb_{dur}$, which will be specified in Section 3.4.1.

3.2.5. Active time.

The working hours of our lab are from 09:00 to 23:00. Based on the testing environment of the lab and the active time feature of a malware, the data flow that is in connection outside working hours of the lab is more possibly to be considered abnormal, and the probability is defined as $TrojanFProb_{ct}$, which will be specified in Section 3.4.1.

3.3. Connection maintenance

When no command sent to the malware and all commands are accomplished on the control side, the connection enters into the free stage [22]. For maintaining this state, the malware and control side would periodically send probing packets to monitor whether the other side is online. The periodically probing packet is called heartbeat packet. On this stage, the average length and the amount of upstream and downstream packets are very small.

3.3.1. The analysis of heartbeat packets.

A heartbeat packet is a kind of user-defined, fixed-length, and periodically sent packet in the web data flow [23]. For example, t, during a heartbeat cycle, there are always two 273-byte length heartbeat packets from the server side and three 108-byte length heartbeat packets from the control side. But the amount of heartbeat packets does not confirm this rule strictly. In Figure 5, the amount of heartbeat packets and the size of sending and receiving data of the malware PoisonIvy [19] are shown, which are the packets from the training data. The absolute value of Y-axis means the length of sending (negative value) or receiving (positive value) packets, and X-axis means the time of sending or receiving packets. There are two types of packets. One is the ACK (acknowledgment character) packet without data, the others are probing and response packets including some data with over 58 bytes. By the influence of network environment, the system in the malware does not send ACK packet every time to the control side. Thus, the amount of upstream and downstream packets in the heartbeat packet clusters is influenced heavily by the network environment. However, there are probing and response data with the same structure in the upstream and downstream data in every heartbeat packet, so the data in application layer loaded in upstream and downstream data packets do not change and it has small amount. This pattern can be obviously observed in Figure 5. The same pattern can be acquired in the analysis of heartbeat packets from other malware abnormal traffic.

DOI: 10.1002/sec

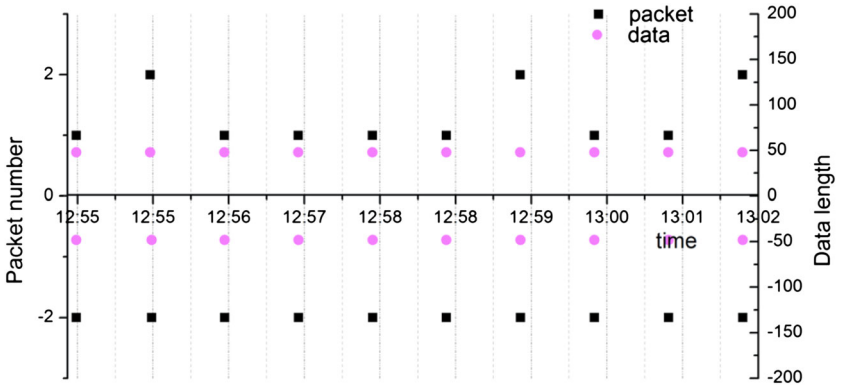

Figure 5. The time distribution of heartbeat packets for malware PoisonIvy.

As a result, we divide the packets into clusters according to the time distribution of upstream and downstream data flows. More specifically, two packets, sent or received, with time interval of less than T (1 s) most possibly belong to the same data packet cluster. The ones that belong to different clusters are in the time interval of over 5 s. Here, we acquire the heartbeat package cluster collection, which is in time interval of less than 1 s through Equation 4.

$$HP = \{H_1, H_2, \cdots, H_n\} \tag{4}$$

In Equation 4, HP refers to the collection of malware heartbeat package; n refers to the number of clusters; $H_i(i = 1, 2, \cdots, n)$ refers to the first n heartbeat packet cluster that connects the heartbeat maintenance stage; the time interval of the packets in the same cluster is less than 1 s and the one in the other cluster is over 5 s.

Heartbeat packets generally have the similar pattern in time interval, besides, most malwares have same time intervals between their heartbeat packets. However, to escape from detection, there are still some Trojans that adopt the specific algorithm to make the time interval change regularly. Though these Trojan's heartbeat packets don't have obvious periodical pattern, the traffic feature of upstream and downstream data in clusters never changes. As a result, we can detect the heartbeat packets clusters according to the variation of their data length, and the variation can be formulated by the standard deviation of clusters' data length. The heartbeat packets are all short packets, because they are just used to keep the connections instead of transmitting data. So there is no much difference between the packets length of the upstream and downstream. Besides, when servers received heartbeat packets from clients, they send echo packets to the clients. As a result, in a heartbeat clusters, the upstream and downstream have similar packets number and data length. So when we compute the variation of the packets clusters, the upstream and the downstream have the same weight. σ_{HD}, the variation of heartbeat package loaded traffic in application layer is calculated as Equation 5.

$$\sigma_{HD} = \frac{1}{2}(\sigma_{HUD} + \sigma_{HDD}) \tag{5}$$

where σ_{HUD} is the standard deviation of the upstream change of heartbeat package clusters and σ_{HDD} is the standard deviation of the downstream. The data flow that meets the condition of $\sigma_{HD} < 2$ and $n > 10$ has the heartbeat maintenance characteristic, and the abnormal traffic probability with this characteristic is defined as $TrojanFProb_{hp}$, which will be specified in Section 3.4.1.

3.4. Abnormal port judgment

The network behavioral features discussed earlier are the abnormal behavioral features that a malware shows in the network activities. But it does not mean that only malware can have these features. Therefore, we use the feature suspicious probability to identify the abnormality of a certain data flow feature.

Web traffic can only stay in one of the three stages mentioned earlier at any one moment and shows the behavioral feature on one communication stage. The same malware could show different features at different time. Therefore, it would lead to high false positive rate and false negative rate if judging from only one result of detecting one item of data flow. To solve this problem, we propose a joint suspicious probability method, which measures different data flow at different time, to detect the malware. The control port of the same malware control side is unchangeable, that is to say, different data flow generated at different time by one malware uses the same destination port. We can calculate the joint suspicious probability of different data flows, which use the same destination port in a period to judge the possibility of this port that is used by attackers to issue attacks.

3.4.1. The calculation of feature suspicious probability.

We further discuss the feature suspicious probability $TrojanFProb_i$ here, which can be obtained from the statistical analysis between a large amount of normal data flow and malware abnormal communication data flow.

$TrojanFProb_i$ is influenced by the proportion of feature i data flow in normal communication data flow, namely, if the proportion is larger, the feature i performs vaguer as abnormal behavioral feature, and $TrojanFProb_i$ value becomes smaller. At the same time, $TrojanFProb_i$ is also related to feature i in the abnormal data flow. If the proportion of feature i data flow in abnormal communication is larger, feature i performs more obvious as the abnormal one, and $TrojanFProb_i$ value becomes larger. More specifically, we define Equation 6

$$TrojanFProb_i = \frac{TrojanNum_i}{\frac{NormNum_i \times TrojanTotal_i}{NormTotal_i} + TrojanNum_i} \tag{6}$$

where $TrojanFProb_i$ is the suspicious probability of feature i, namely, the possibility of feature i data flow being abnormal; $TrojanNum_i$ is the number of abnormal data flow items, which fit feature i in the training set; $TrojanTotal_i$ is the sum of abnormal data flow items in the training set; $NormNum_i$ is the number of normal data flow items, which fit feature i in the training set; and $NormTotal_i$ is the sum of normal data flow items in the training set.

When one item of data flow has one of the aforementioned abnormal behaviors in detecting time, the probability of abnormal data flow is presented as feature suspicious probability. The data flow on operating control stage may have multiple abnormal behavioral features, thus, the suspicious probability is the average value of the suspicious probabilities of these features.

3.4.2. The calculation of port abnormal probability.

In single detection, the suspicious probability of a remote port is defined as Equation 7.

$$TrojanPort_t = 1 - \prod_{i=1}^{m}(1 - TrojanFProb_i) \tag{7}$$

In Equation 7, m is the total number of abnormal features detected on the flows connecting to port t during the detection period. And $TrojanFProb_i$ is the suspicious probability of abnormal feature i. The suspicious probability of a remote port is calculated based on multiple features. The higher the features' suspicious probabilities are, the higher the remote port's suspicious probability is. Besides, if the suspicious probability of one feature is 1 (100%), the suspicious probability of the remote port is 1 (100%) according to Equation 7.

In time T, the probability of detecting abnormal port is described as Equation 8, where dec_times is the sum of this detecting times during the detecting period T. The larger is the $TrojanPort$ value, the greater is the possibility of being an abnormal port.

$$TrojanPort = \frac{1}{dec\_times}\sum_{t}^{T} TrojanPort_t \tag{8}$$

4. PARAMETER SETTING AND MODEL VERIFICATION

In this section, we describe the detailed testing environment in our lab for prototype model verification. There are eight known Trojan programs installed randomly on seven of the 15 computers. Network communication data are captured on the image port of switches in the lab. We choose 737 items of normal data flow randomly. A total of 74 items are in tentative connection stage; 547 items are in operating control stage; and 116 items are in connection maintaining stage. A total of 234 items of abnormal data flow are chosen. Among them, the numbers in each stage are 93, 67, and 74 respectively. Table II shows the parameter settings of the data flow features.

In different network environment, there are distinctions in the statistic results of different thresholds and data flow features and so is the possibility of abnormality. As a conclusion from the statistics result, if judging abnormal data flow from only one behavioral feature, the false positive rate is relatively high. For example, in the feature statistics process of normal data flow, we find that one IP address in the LAN keeps on trying to connect the remote port 130.158.6.56:80. Through artificial detection, these connections are normally sent by client PacketiX VPN Client Manager. By using the proposed algorithm and the statistics joint detection probability in 24 h, the port suspicious probability value is low. As a result, the data flow has only one suspicious behavior feature.

The statistics results of the data flows from the control side of eight malwares in 24-h detection are shown in Table III.

We can find from the detection results that, in 24 h, the monitoring port of the same malware always shows different behavioral features. By setting specific detection period, and taking comprehensive consideration of the detection results in different time intervals of the period, the final detection method can effectively solve the vague performing problem of malicious communication behavioral feature in one detection period and lower the false positive rate.

The software gwgirl [19] shows the same feature as the malware on the tentative connection stage and connection maintaining stage. However, its online method is a passive connection model. When establishing a connection, it is

Table II. Parameter settings of the data flow features.

Behavior		Normal	Malware	
Phase	Feature	Session	Session	$TrojanFProb$(%)
Establish	Connect	23	79	73.2
	Flow	103	61	82.9
	Packet	135	59	78.1
Operation	Shell	97	41	77.5
	Duration	31	65	94.5
	Time	37	54	92.3
Idle	Heartbeat	19	67	84.7

DOI: 10.1002/sec

the control side, not gwgirl, that sends a connection request to malware. It does not possess the characteristics of a tentative connection. The connection is passive and the data do not need maintenance after delivery. Thus, it does not possess the characteristics of heartbeat packets.

Magic, gwboy, and Huigezi [19] also do not possess the characteristics of heartbeat packets on the connection maintaining stage. Actually, these software do not maintain the same connection on the free stage. After establishing connection and verifying the information, they will receive the reset packets sent by the control side. The malware will re-establish the connection after some time and repeat this connection process. The phenomenon possesses the characteristics of the tentative connection.

Because part of hosts may be shut down while detected, or the malware has not been controlled by the control side, some of the malwares that own network communication behavior features are not obvious. However, as the statistics results shown in Table III, almost every malware possesses over three behavior features in 24 h, the same as our proposed malware detection model DTrojan.

On the test stage, many network games possess some characteristics of network behavior, so they may be also detected as Trojans. But the traffic generated by these games, which has large download traffic, is unique from that of malware. The control side of malware does not need to send much data to malware, so we can remove the game software judging from the downstream traffic. In the experiment, it is found that the average downstream traffic of all Trojans is less than 2 KBps. As a result, we set the threshold value as 2 KBps to avoid the mistake made from network games.

5. PROTOTYPE SYSTEM DEVELOPMENT

Based on the proposed malware detection system DTrojan described earlier, we have developed the prototype DTrojanDemo under operating system Ubuntu-11.10, which has been used to detect the abnormal traffic and find suspicious ports of the LAN.

Figure 6 shows the framework of DTrojanDemo. DTrojandemo mainly consists of three modules: flow preparation module, flow detection module, and detection results query module. The first two modules are written in C and Mysql, and the last module is realized by JSP and Apache, which will be explained in details as follows.

5.1. Flow preparation module

Three functions are realized in this module: packet capture, flow regroups, and flow storage. DTrojanDemo uses PF_RING [24] to capture TCP packets in the switch image port of the LAN. It regroups and identifies every TCP packet according to five-tuples source IP address, source port number, destination IP address, destination port number, and detecting time.

The storage structure of data flow in random access memory is shown in Figure 7. The data flow header con-

Table III. Parameter settings of the data flow features.

Malware	Port	Try connect	Flow	Packet	Shell	Duration	Time	Heartbeat
Pchshare	80	Y	Y	Y	Y	N	Y	Y
PoisonIvy	3460	Y	Y	Y	Y	Y	Y	Y
Magic	9999	Y	Y	Y	Y	N	Y	Y
Gwboy	90	Y	Y	Y	N	Y	N	N
Gwgirl	6267	N	Y	Y	N	Y	Y	N
Regoice	8181	Y	Y	Y	Y	Y	N	Y
Netsys	6678	Y	Y	Y	N	Y	Y	Y
Huigezi	9000	Y	Y	Y	Y	Y	Y	N

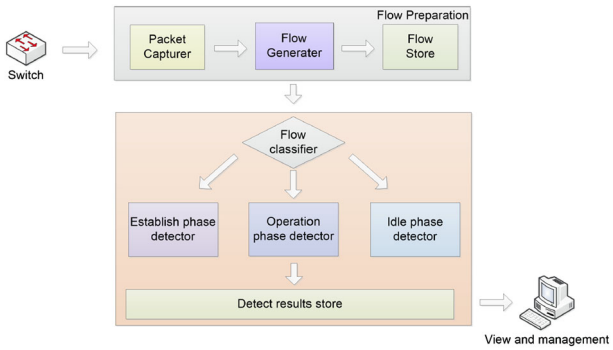

Figure 6. Framework of DTrojanDemo.

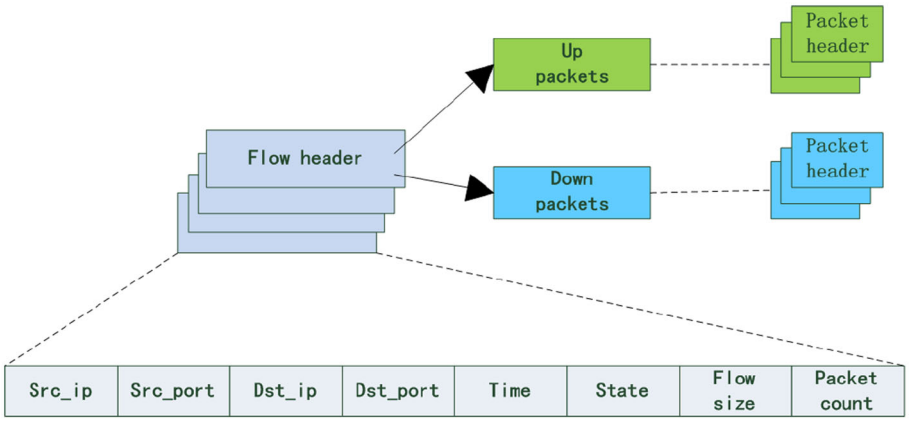

Figure 7. Storage structure of data flow.

tains source IP address, source port number, destination IP address, destination port number, capturing time (storage-in time), flow statement, flow size (including upstream and downstream), number of packets (including upstream packets and downstream packets), and so on. Every item of data flow includes two packet chains, namely, upstream packet chain and downstream packet chain. As a result of long period in detecting module and large amount of data flow, the finished and time-out sessions [25] are saved in the database. DTrojanDemo only detects communication behavior of data flow, but for content detection, only packet header data of session unit are saved, without load data of packets. The data flows stored in database are classified according to their local IPs. Not all the data flows have the detecting condition, such as communication in short time or large downstream and small upstream data, which need not be stored in database. The data flow, which satisfies one of the following conditions, can be stored in database.

(1) Data flow that has not been connected successfully.
(2) Data flow whose connection duration is over 10 min and upstream traffic is no less than three times of downstream traffic.
(3) Data flow whose connection duration is over 10 min and the average length of both upstream and downstream packets is smaller than 110 bytes.

5.2. Flow detection module

The detecting duration of DTrojanDemo is set to 1.5 h, that is, DTrojanDemo would detect packets in 1.5 h from a certain destination port. The network communication data of a host (source IP address) are used as an example. The procedure is described as follows.

(1) Data flow sorter regroups the 1.5 h data according to destination IP address and destination port number.
(2) Extract the failed connection data flows, which have the same source and destination IP addresses and destination port number, in $Port_i$ to make up the tentative connection data flow. Then it is sent to behavior detector of connection establishment stage.
(3) Calculate the average upstream and downstream packet lengths of the surplus data flow. The data flow whose traffic is less than 110 bytes would be sent to behavior detector of free stage, and the others are sent to behavior detector of the operating stage.
(4) According to the detection result and calculating method in Section 3.4.2, calculate the abnormal probability of $Port_i$ in the detecting process.

5.3. System management module

When network administrator inquires the detection results, the system would display the details of the suspicious probability of each suspicious port and the abnormal behavior of a certain suspicious port in this period.

Under default circumstance, this system would calculate the suspicious port probability of the last 24 h at 8:00 AM every day and sends the detection report to the email address of network administrators. Figure 8 shows information provided by the user interface of the prototype system.

In order to overcome the high rates of false positives and false negatives of signature-based malware detection and host-based detection, in this paper, we propose a new network behavior evidence chain-based malware detection method named DTrojan. The DTrojan analyzes the network behavior feature of data flow to find the abnormal ones. Then it comprehensively analyzes several items of data flow with the same destination port in a period to calculate the suspicious probability of malware monitoring port. The detecting target is the behavior feature of data flow, not communication content or a single packet. Thus, the detection efficiency is improved and the influence from content encryption and protocol masquerading is avoided.

The feature threshold value and suspicious probability in the model are specified in this paper. Though the

DOI: 10.1002/sec

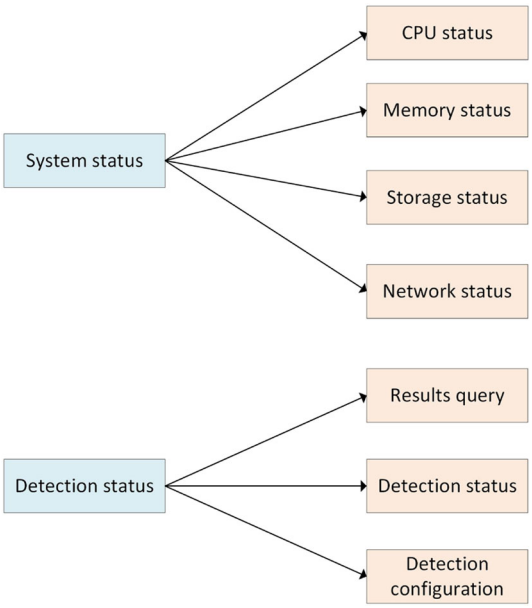

Figure 8. Information provided by user interface of DTrojanDemo.

model has passed through large amount of data statistics and experimental verification, the detection results would be different in threshold value and suspicious probability with the traffic from different network environment such as LAN for educational, LAN for family, and LAN for internet cafe. Therefore, we need to find better method under different network environment. Besides, a large amount of P2P traffic exists in the network communications, which occupy much cache space and detection time. A new filter method is needed to identify the P2P traffic before data flow storage.

6. CONCLUSION

In this paper, we propose a new malware detection technique DTrojan, which is based on network behavior characteristics and utilizes the classification algorithm from the data mining to analyze the real traffic data of a LAN to identify the malware. The DTrojan focuses on the malware's communication flows instead of the communication contend and single packet, which detects not only the known malware but also the unknown and new malwares and malwares using encrypted communication content efficiently. By using behavioral characteristics to detect malware, this technique can satisfy the needs for a large database of signatures to identify each known malware. Another important benefit is that the detecting rules can be adjusted and optimized according to the real network environment, detecting better results in different network environments. Besides, this technique can also be expanded to detect other malware, such as worms, virus, and adware not only detecting spyware.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper. This paper is supported by the Chinese National Natural Science Foundation (No. 61373006), the Foundation of Nanjing University of Posts and Telecommunications (No. NY212059), and A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD).

REFERENCES

1. Roesch M et al. Snort: lightweight intrusion detection for networks, In *LISA*, Seattle, Washington, 1999, volume 99; 229–238.
2. Murphy KP. Naive bayes classifiers, 2006.
3. Deng PS, Wang JH, Shieh WG, Yen CP, Tung CT. Intelligent automatic malicious code signatures extraction, In*Proceedings of IEEE 37th Annual 2003 International Carnahan Conference on Security Technology*, Phuket, 2003; 600–603.
4. Wang C, Sun L, Wei J, Mo X. A new trojan horse detection method based on negative selection algorithm, In *Proceedings of 2012 IEEE International Conference on Oxide Materials for Electronic Engineering (OMEE)*, Lviv, Ukraine, 2012; 367–369.
5. Fiskiran AM, Lee RB. Runtime execution monitoring (rem) to detect and prevent malicious code execution, In *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD)*, San Jose, California, 2004; 452–457.
6. Al-Saadoon GMW. Authentication and virus detection enhancement for client and server applications. *Journal of Computing and Information Sciences* 2011-08; **2**(8): 390–295.
7. Moser A, Kruegel C, Kirda E. Limits of static analysis for malware detection, In *Proceedings of Twenty-Third Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, Florida, 2007; 421–430.
8. Wu N, Qian Y, Chen G. A novel approach to trojan horse detection by process tracing, In *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control (ICNSC'06)*, Ft. Lauderdale, Florida, 2006; 721–726.
9. Schmall M. *Heuristic techniques in av solutions: An overview*, 2002. Available from: http://www.securityfocus.com/infocus/1542 [Accessed on 23 April 2013].
10. Luo H, MU D, DAI G, YUAN Y. Research on communication techniques for port recall trojan horse. *Microelectronics & Computer* 2006; **2**: 99–101.

11. Kalafut A, Acharya A, Gupta M. A study of malware in peer-to-peer networks, In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, Pisa, Italy, 2006; 327–332.
12. Kumar S, Pant D. Detection and prevention of new and unknown malware using honeypots. *CoRR* 2009; **1**(2): 56–61.
13. Moskovitch R, Elovici Y, Rokach L. Detection of unknown computer worms based on behavioral classification of the host. *Computational Statistics & Data Analysis* 2008; **52**(9): 4544–4566.
14. Moskovitch R, Nissim N, Englert R, Elovici Y. Active learning to improve the detection of unknown computer worms activity, In *Proceedings of 11th International Conference on Information Fusion*, Cologne, Germany, 2008; 1–8.
15. Bianchi A, Shoshitaishvili Y, Kruegel C, Vigna G. Blacksheep: detecting compromised hosts in homogeneous crowds, In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, Raleigh, NC, 2012; 341–352.
16. Xu Z, Chen L, Gu G, Kruegel C. Peerpress: utilizing enemies' P2P strength against them, In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, Raleigh, NC, 2012; 581–592.
17. Gu G, Porras P, Yegneswaran V, Fong M, Lee W. Bothunter: Detecting malware infection through ids-driven dialog correlation, In *Proceedings of 16th USENIX Security Symposium*, Boston, MA, 2007; 1–16.
18. Yoda Kunikazu, Etoh Hiroaki. Finding a connection chain for tracing intruders, *Proceedings of the 6th European Symposium on Research in Computer Security,* Toulouse, France, 2000; 191–205.
19. Pu Y, Chen X, Cui X, Shi J, Guo L, Qi C. Data stolen trojan detection based on network behaviors. *Procedia Computer Science* 2013; **17**: 828–835.
20. Ye M, Xu K, Wu J, Po H. Autosig-automatically generating signatures for applications, In *Proceedings of Ninth IEEE International Conference on Computer and Information Technology (CIT'09)*, Xiamen, China, 2009; 104–109.
21. Jia WL, Xue Q, Sun JZ. Distributed port recall (DPR) attacks and its detection, In *Proceedings of International Conference on Machine Learning and Cybernetics*, Xian, China, 2003; 2346–2350.
22. Borders K, Prakash A. Web tap: detecting covert web traffic, In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, Washington, DC, 2004; 110–120.
23. Li F, Yu X, Wu G. Design and implementation of high availability distributed system based on multi-level heartbeat protocol, In *Proceedings of IITA International Conference on Control, Automation and Systems Engineering (CASE)*, Bangalore, India, 2009; 83–87.
24. Ntop. High-speed packet capture, filtering and analysis. Technical Report.
25. Sen S, Spatscheck O, Wang D. Accurate, scalable in-network identification of P2P traffic using application signatures, In *Proceedings of the 13th International Conference on World Wide Web*, New York, NY, 2004; 512–521.