

國立交通大學

網路工程研究所

碩士論文

雲端計算環境下基於網路行為之殭屍網路偵測機制

Behavior-based Botnet Detection in Cloud Computing Environments

研究生：蔡禮陽

指導教授：王國禎 博士

中華民國一百年六月

雲端計算環境下基於網路行為
之殭屍網路偵測機制

Behavior-based Botnet Detection in Cloud Computing
Environments

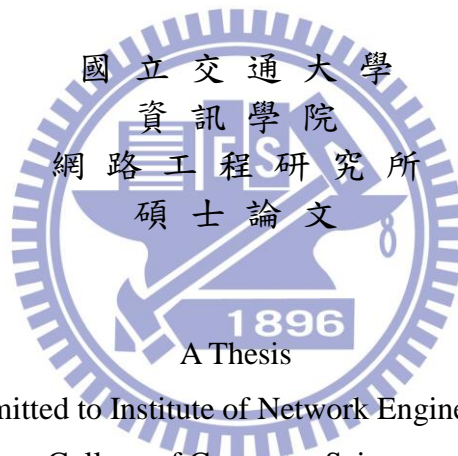
雲端計算環境下基於網路行為
之殭屍網路偵測機制

研究生：蔡禮陽

Student : Li-Yang Tsai

指導教授：王國禎

Advisor : Kuo Chen Wang



Submitted to Institute of Network Engineering
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master
in
Computer Science

June 2011

Hsinchu, Taiwan, Republic of China


中華民國一百年六月

雲端計算環境下基於網路行為 之殭屍網路偵測機制

學生：蔡禮陽 指導教授：王國禎 博士

國立交通大學 資訊學院 網路工程研究所

摘要



殭屍網路在近幾年非常盛行，造成經濟及隱私上的安全危害及分散式阻斷攻擊等網路犯罪的問題。傳統的字串比對偵測方法在殭屍網路的偵測上容易發生誤判或漏判的情況。為了解決這個問題，在本論文中，我們提出雲端計算環境下基於網路行為之殭屍網路偵測機制，簡稱 BBDC，來分析網路流量以偵測殭屍網路。我們根據錄製的各網路封包之行為來做殭屍網路之分析與偵測。BBDC 分成五個階段，第一個階段是利用殭屍網路的特性來過濾掉不需要檢查的封包。第二個階段則是取出封包流量的特徵。第三個階段則將已經過濾完剩下的待測封包流量切割成多個相同大小的資料量送入雲端系統的多個虛擬機器進行殭屍網路檢測。第四個階段及第五個階段則是透過模糊識別對 DNS 封包及 TCP 封包的行為來進行殭屍網路偵測。當待測封包

被確認為殭屍網路的流量，本機端的電腦以及在雲端內的伺服器群可以預防殭屍網路的危害經由儲存在資料庫的殭屍網路的相關資料。為了評估此方法的有效性，我們收集了真實殭屍網路流量及校園宿舍正常流量來評量我們的方法。實驗結果顯示，我們提出的 BBDC 對於殭屍網路的流量辨識正確率高達 95.83%，且對於正常網路流量只有 0% ~ 3.453% 的誤判率。此外，我們引入雲端計算的技術，使用五台虛擬機器去進行殭屍網路的流量偵測，與只在本機端的殭屍網路偵測相比，我們提升 4.73 倍的殭屍網路偵測速度。此證明我們提出的偵測機制可藉由雲端計算環境資源達到快速偵測殭屍網路之結果。

關鍵詞：基於行為比對、殭屍網路偵測、雲端計算環境、模糊識別、基於字串比對。



Behavior-based Botnet Detection in Cloud Computing Environments

Student : Li-Yang Tsai

Advisor : Dr. Kuochen Wang

Department of Computer Science

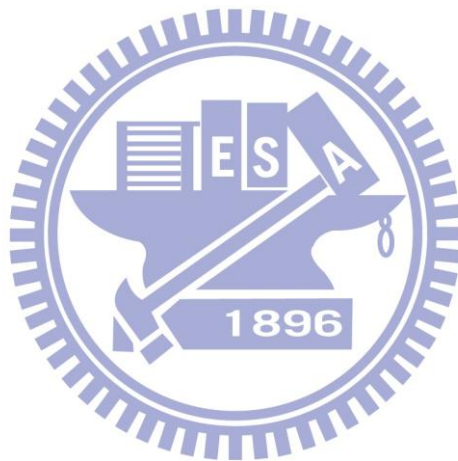
National Chiao Tung University

Abstract

In recent years, botnets become a major issue to Internet security; however, existing string signature-based matching methods usually lead to high false positive rates (FPR) and low true positive rates (TPR) for botnet detection. In this paper, we proposed a *behavior-based botnet detection* mechanism in cloud computing environments (BBDC). Our BBDC algorithm is divided into five stages: (1) *traffic reduction*: removing unwanted packets from an input trace for speeding up bot detection; (2) *feature extraction*: extracting features from the reduced input trace; (3) *traffic partitioning*: dividing the reduced input trace into pieces for a cloud-based system to detect botnets concurrently; (4) *DNS phase*: extracting botnet DNS features to detect bots; (5) *TCP phase*: extracting TCP request and response features to detect bots. Since stage four and five consume almost 90% of the total execution time in our design, we dispatch reduced input traces to the cloud to speed up botnet detection. In order to achieve a high detection rate, we utilize fuzzy pattern recognition for botnet detection in DNS and TCP phases. Once bot activities are identified from the input trace, local hosts and servers in the cloud will be alerted to avoid bot related IP addresses or domain names (DNs). Experimental results show that the proposed BBDC can achieve high TPR and low FPR. Furthermore, the proposed cloud-based

botnet detection system with five virtual machines is 4.73 times faster than a host-based system.

Keywords: behavior-based, botnet detection, cloud computing environment, fuzzy pattern recognition, signature-based.



Acknowledgements

Many people have helped me with this thesis. I deeply appreciate my thesis advisor, Dr. Kuochen Wang, for his intensive advice and guidance. I would like to thank all the members of the *Mobile Computing and Broadband Networking Laboratory* (MBL) for their invaluable assistance and suggestions and the *Network Benchmarking Lab* (NBL) for their botnet samples. The supports by the National Science Council under Grant NSC 99-2221-E-009-081-MY3 and Microsoft are also gratefully acknowledged.

Finally, I thank my family for their endless love and support.



Contents

Abstract (in Chinese)	i
Abstract	iii
Contents	vi
List of Figures	viii
List of Tables	ix
Abbreviations List	1
Chapter 1 Introduction	2
Chapter 2 Background and Related Work	4
2.1 Overview of botnet behaviors	4
2.2 Related work	6
Chapter 3 Proposed Behavior-based Botnet Detection Algorithm in Cloud Computing Environments	8
3.1 Problem statement.....	8
3.1.1 The main problem.....	8
3.1.2 The sub-problems	8
3.2 Design of a botnet detection algorithm.....	9
3.3 Traffic reduction.....	11
3.4 Feature extraction.....	12
3.4.1 Feature extraction from DNS packets.....	12
3.4.2 Feature extraction from TCP connection packets.....	12
3.5 Fuzzy pattern recognition	13
3.5.1 DNS phase	14
3.5.2 TCP phase.....	17
3.5.3 Observation.....	20

Chapter 4 Performance Evaluation23

 4.1 Traces collection23

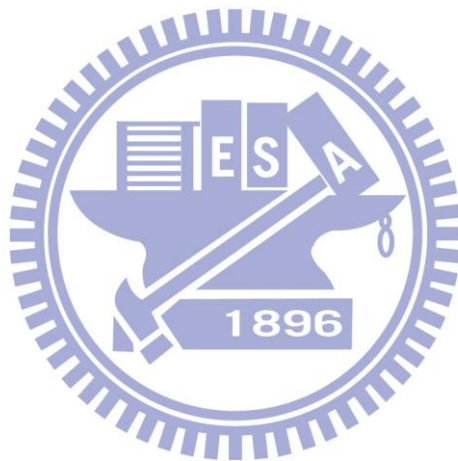
 4.2 Test results of botnet traces24

Chapter 5 Conclusion29

 5.1 Concluding remarks29

 5.2 Future work.....29

References30

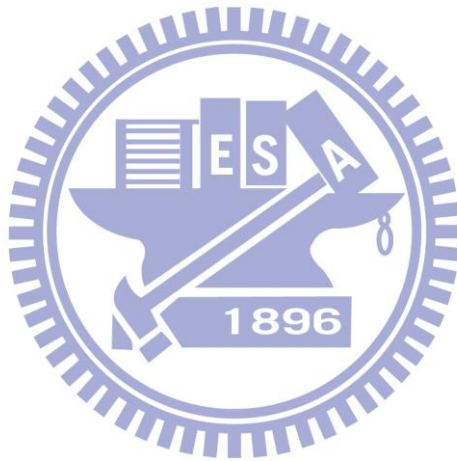


List of Figures

Figure 1. Botnet behaviors scenario: infection and attack phases .	5
Figure 2. Proposed behavior-based botnet detection in a cloud computing environment.	10
Figure 3. The procedure in traffic reduction stage.	11
Figure 4. The distribution of botnet DNS query packets. [x axis: seconds; y axis: number of DNS query packets].	12
Figure 5. The distribution of botnet TCP request packets. [x axis: seconds; y axis: number of TCP request packets].	13
Figure 6. The DNS phase of the proposed botnet detection algorithm.	14
Figure 7. The TCP phase of the proposed botnet detection algorithm.	14
Figure 8. Fuzzy max membership principle in DNS phase.	15
Figure 9. Fuzzy max membership principle in TCP phase.	17
Figure 10. Experimental environment for botnet traces collection.	23
Figure 11. Total execution time (sec) for 950 traces using various number of server instances (SIs) in Windows Azure cloud	25
Figure 12. False positive bots distribution with respect to bot features in DNS phase.	26
Figure 13. False positive bots distribution with respect to bot features in TCP phase.	26
Figure 14. False negative bots distribution for different bot types.	27

List of Tables

Table 1. Botnet traces statistics and false negative rate.	24
Table 2. Normal traces statistics and false positive rates.	24
Table 3. Comparison of different behavior-based botnet detection methods.	28



Abbreviations List

The list contains the main abbreviations used throughout this thesis.

BBDC: Behavior-based botnet detection in cloud computing environments

C&C: Command and control

DDoS: Distributed denial of service

DN: Domain name

DNS: Domain name system

FN: False negative

FNR: False negative rate

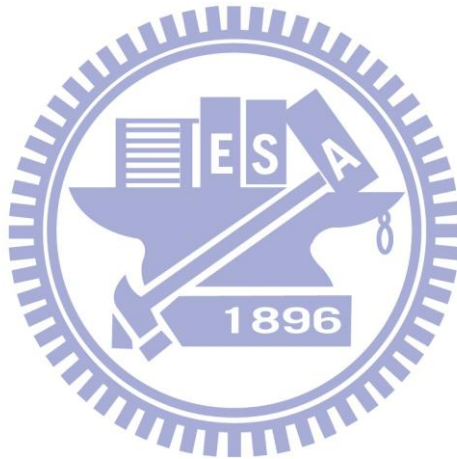
FP: False positive

FPR: False positive rate

SI: Server instance

TP: True positive

TPR: True positive rate



Chapter 1

Introduction

In recent years, Botnets have become new threats to Internet. They can duplicate themselves and spread to other hosts quickly. Once a host is compromised, user privacy data on the host may be stolen and it may result in severe damages (e.g., finance and security).

Nowadays, most anti-virus systems and botnet detection methods primarily use string signature-based methods to detect well-known botnets [2][7][8][9][10]. Although a signature-based solution may have high detection results, it has the following drawbacks. First, since botnets may change their behaviors (e.g., communication, attacking, etc.) from time to time, the string signature-based pattern matching methods have difficulties to deal with varying behaviors of botnets [1]. Second, a bot is able to evade signature-based detection easily by using techniques such as code obfuscations and mutations [12]. Finally, the string signature-based database must be maintained by humans. On the contrary, behavior-based solutions try to identify bot activities by observing particular bot network behaviors. In addition, a behavior-based system does not need to maintain a signature database to detect bots. Furthermore, behavior-based solutions are able to perform similar detection rates to signature-based solutions [1].

In this paper, we propose a novel technique which identifies botnet behaviors by two phases botnet matching. The two phases includes: the DNS phase and the TCP phase. The DNS phase focuses on analyzing botnet DNS queries since bots activities often start with DNS traffic. The TCP phase involves on TCP request and response

packets. Moreover, we raise our detection rates by utilizing fuzzy pattern recognition to identify bot behaviors in the DNS and TCP phases. DNS and TCP phases' features have been extracted from observed bot behaviors. These features can be fed into the fuzzy pattern recognition module to calculate membership values to identify possible bot activities.

By observing the activities of collected network traces, we can identify bot activities based on extracted bot features and conclude that the corresponding input trace is compromised by a bot. Finally, we use a Windows Azure's cloud computing platform [11] to speed up bot detection. Each server instance in the cloud can handle any phase of the proposed two phases botnet matching algorithm to determine if an input trace is bot or not. Once identifying bots from the input trace, local hosts and servers in the cloud will be alerted to be aware of the bot related IP addresses or domain names.

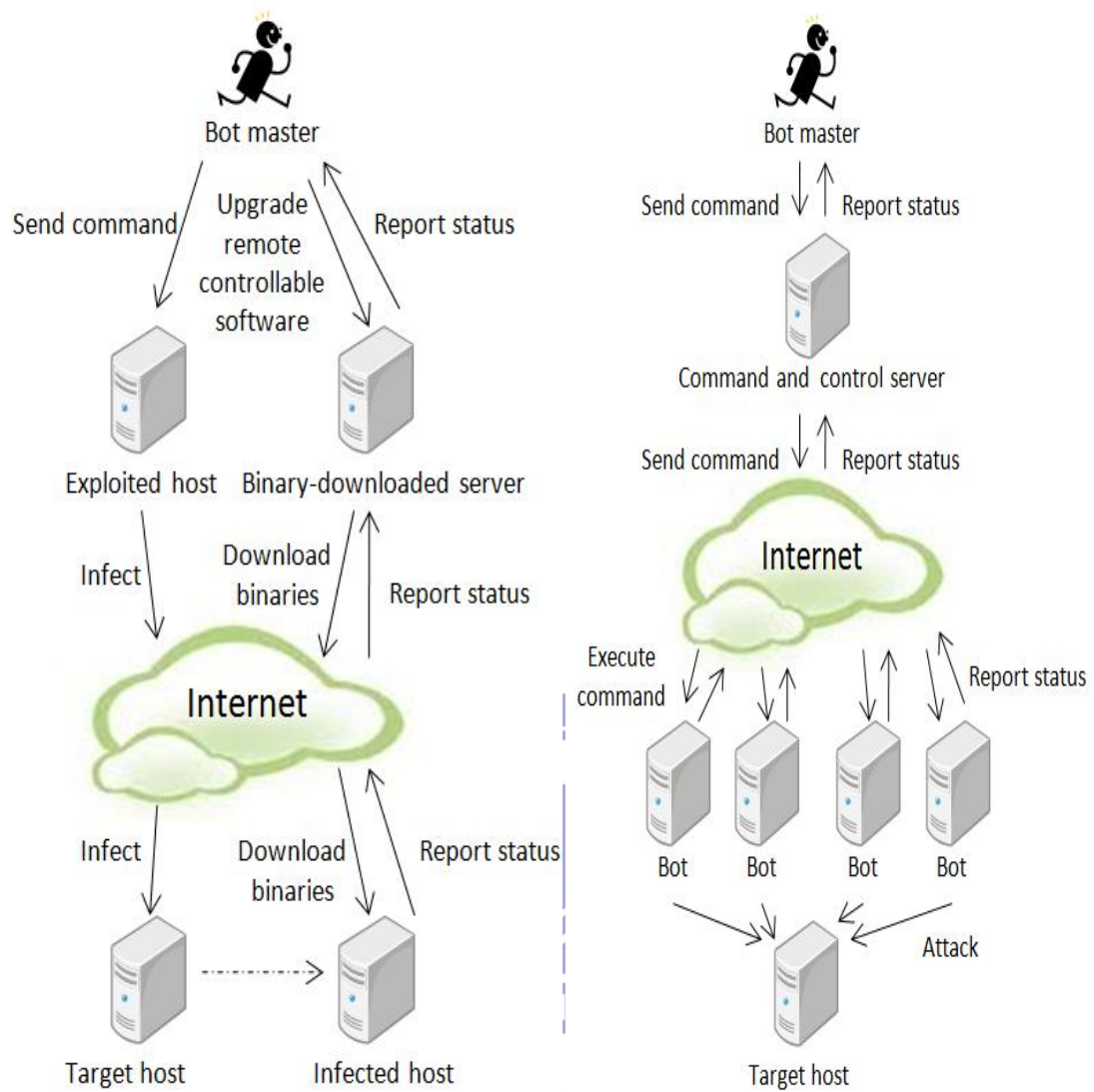
The rest of this paper is organized as follows. Chapter 2 briefly reviews related work. Chapter 3 details the proposed approach for cloud-based botnet detection. Chapter 4 presents the evaluation results of the proposed algorithm using real-world botnet traces. Finally, Chapter 5 gives concluding remarks and future work.

Chapter 2

Background and Related Work

2.1 Overview of botnet behaviors

The scenario of botnet behaviors can be classified into two phases: the *infection phase* and the *attack phase*, as shown in Figure 1 [1]. In the infection phase, as shown in Figure 1(a), a bot master tries to break into a victim's host and makes it become a bot. There are many methods to break into a host, such as exploiting the host vulnerabilities and by social engineering of divulging confidential information [13]. Once the intrusion is successful, the infected host sends its status to the bot master and tries to install remote controllable software, which can be downloaded from a binary-downloaded server. The binary-downloaded server is responsible for reporting infected hosts status, error log and receiving upgraded software. In the attack phase, as shown in Figure 1(b), a bot master sends commands to compromised bots to ask bots to collect valuable information, report botnet status, and launch attacks to target hosts.



(a) Bot master infects a target host. (b) Bot master sends commands to infected hosts (bots) to initiate attacks.

Figure 1. Botnet behaviors scenario: infection and attack phases [1].

2.2 Related work

Park et al. [4] proposed clustering common semantic patterns for botnet detection. This paper presents a behavior-based automated approach to generate semantic patterns for botnet detection. It uses static analysis to characterize bot behaviors and uses hierarchical clustering of the resulting semantic patterns from a set of bot programs. The detection requires a pattern matching to compute a matching score. If the matching score exceeds a pre-defined decision threshold, the new suspicious bot program is an instance of this bot malware class associated with the pattern. The goal is to identify common semantic behaviors of bots. It may miss some malicious behaviors that involve intentional or non-intentional program obfuscation.

Yu et al. [5] proposed online botnet detection based on an incremental discrete Fourier transform method to detect botnets. It monitors botnet activities in an online way. The authors define the concept of “feature streams” to describe raw network traffic. If some feature streams show high similarities of tested traces, the corresponding hosts will be regarded as suspected bots which will be added into the suspected bot hosts set. Since the authors focus only on the overall input traffic, the detection system may handle too many data in the same time that may cause the detection rate no good enough.

The major differences of our proposed method with [4][5] are that we define the DNS and TCP phases and then detect botnets by differential features in each phase. Consequently, the false positive rate can be reduced compared to [4][5], because [4][5] use feature streams or semantic patterns to match the overall input traffic, not by each phase. They may lose some suspicious details in each phase. The proposed method can improve the false positive rate and false negative rate of [4][5].

Wang et al. [1] proposed a fuzzy pattern-based filtering algorithm for botnet detection. The proposed method extracts bot features first and then recognizes botnets based on collected bot behaviors. The *fuzzy pattern recognition* stage has two phases. The *DNS phase* analyzes features of DNS packets. If a domain name is determined to be malicious, the corresponding DN and its associated IP addresses will be marked without going to the next phase. The *TCP connection phase* analyzes features of TCP connection packets. The associated IP addresses will be marked if TCP connection packets are determined to be malicious. Note that the detection rate and false positive rate can be further improved if more bot features are used. Some false positive cases are due to that the algorithm does not handle well on Microsoft Update related traffic. For the detection rate, the problems of this algorithm are that it only focuses on periodical DNS query traffic and similar TCP packet payload sizes to detect bots.

In our approach, we employ more bot features into our proposed algorithm in contrast to Wang et al. to enhance the detection rate and the false positive rate. Since more bot features from DNS and TCP phases can be extracted to represent bot behaviors, the proposed algorithm can be more precisely for botnet detection.

Chapter 3

Proposed Behavior-based Botnet

Detection Algorithm in Cloud

Computing Environments

3.1 Problem statement

3.1.1 The main problem

Given a network packet trace, the goal of the proposed solution is to identify whether the packet trace is generated by bot activities or not. To overcome the drawbacks of string signature-based methods mentioned in Chapter 1, we propose a novel behavior-based botnet detection method in cloud computing environments.

3.1.2 The sub-problems

- *Traffic reduction*

Since there are some bot-unrelated data in the collected input traces, we may filter out these data to speed up botnet detection. Bot behaviors always involve specific network operations. Therefore, we may retain packets related to these operations.

- *Traffic partitioning*

Since we want to reduce the total execution time of the proposed algorithm, we use a cloud computing platform to speed up botnet detection. Traffic partitioning divides an input trace into pieces. Then, available server

instances in the cloud can operate on these pieces concurrently using the proposed botnet detection algorithm.

- *Feature extraction*

In our observation, bots always contain specific behaviors that are different from normal user's behaviors. Therefore, we want to detect bots by features extracted from bot behaviors.

- *DNS phase*

The DNS phase focuses on the bot DNS query and response packets since bot activities often start with DNS traffic. We use fuzzy pattern recognition with max membership principle to identify some bot behaviors from DNS traffic in the DNS phase.

- *TCP phase*

A bot master may send and update bot binaries programs to bots. The TCP phase focuses on the TCP request and response packets. We use fuzzy pattern recognition with max membership principle to identify bot behaviors from TCP related traffic in the TCP phase.

3.2 Design of a behavior-based botnet detection algorithm

The proposed *behavior-based botnet detection in cloud computing environments* (BBDC) algorithm is shown in Figure 2. There are five stages in the algorithm: *traffic reduction, feature extraction, traffic partitioning, DNS phase* and *TCP phase*. First, input traffic is passed to the traffic reduction stage by removing bot-unrelated traffic. Second, the feature extraction stage will extract bot features from the reduced input traffic. The packets' related information will be recorded into the database by bot features that we observed. Third, the reduced input traffic will be

divided into several pieces according to available server instances in the cloud. Each server instance in the cloud then runs the proposed two phase bot detection algorithm to detect bots from the received piece. Fourth, in the DNS phase, the detection algorithm checks if any DNS related packets are bot traffic or not based on bot features extracted in the second phase. If there is no bot found in the DNS phase, the piece will be fed into the TCP phase to determine whether any TCP related packets are bot traffic or not. Both the DNS and TCP phases use the proposed fuzzy pattern recognition with max membership principle to match bot behavior states. If the membership value falls into any bot's state, the input trace is identified as a bot.

Once an input trace is identified as a bot, we will record the bot related information in the database, such as DNS and IP addresses. Then the other hosts can be informed of these bots via the alert system.

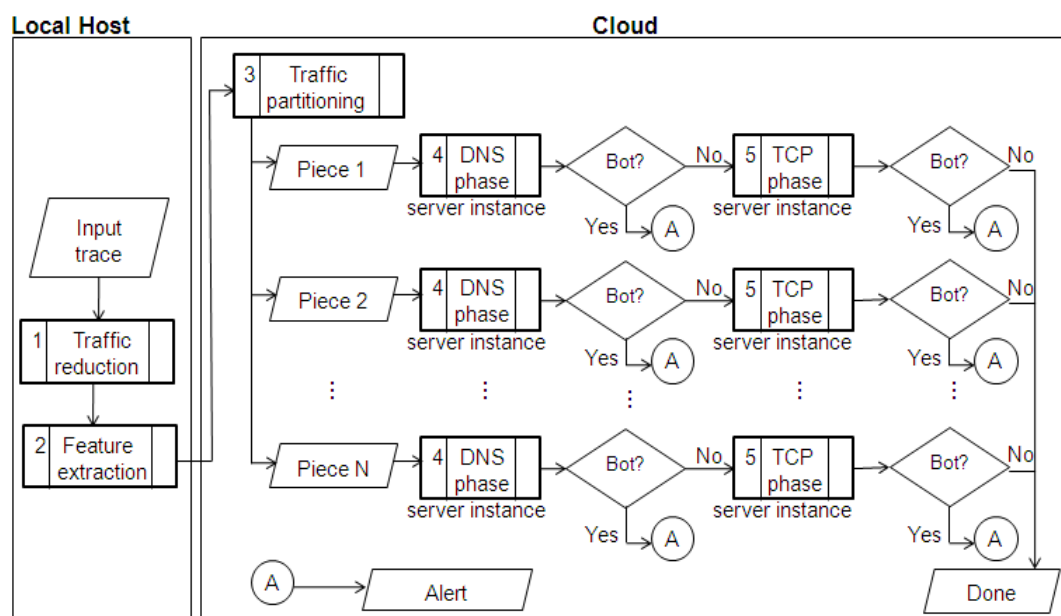


Figure 2. Proposed behavior-based botnet detection in a cloud computing environment.

3.3 Traffic reduction

It is true that a good traffic reduction filter can reduce the data needed to be processed and also increase classification accuracy [1]. That is, by removing these unrelated data, both bot detection time and bot detection accuracy can be improved. Figure 3 shows the procedure in traffic reduction stage. An input trace usually involves several network protocols. In our observations, a bot master may register many domain names and let bots to inquire IP addresses of these domain names. Therefore, bots often send DNS queries to domain name servers frequently to retrieve IP addresses. Then, bots will establish TCP connections to these IP addresses. That is, bot behaviors always involve DNS query/response and TCP request/response packets. Based on these observations, we can filter out packets that are not related to DNS or TCP protocols. The retained packets will then be stored in a database (DB) for feeding into the feature extraction stage.

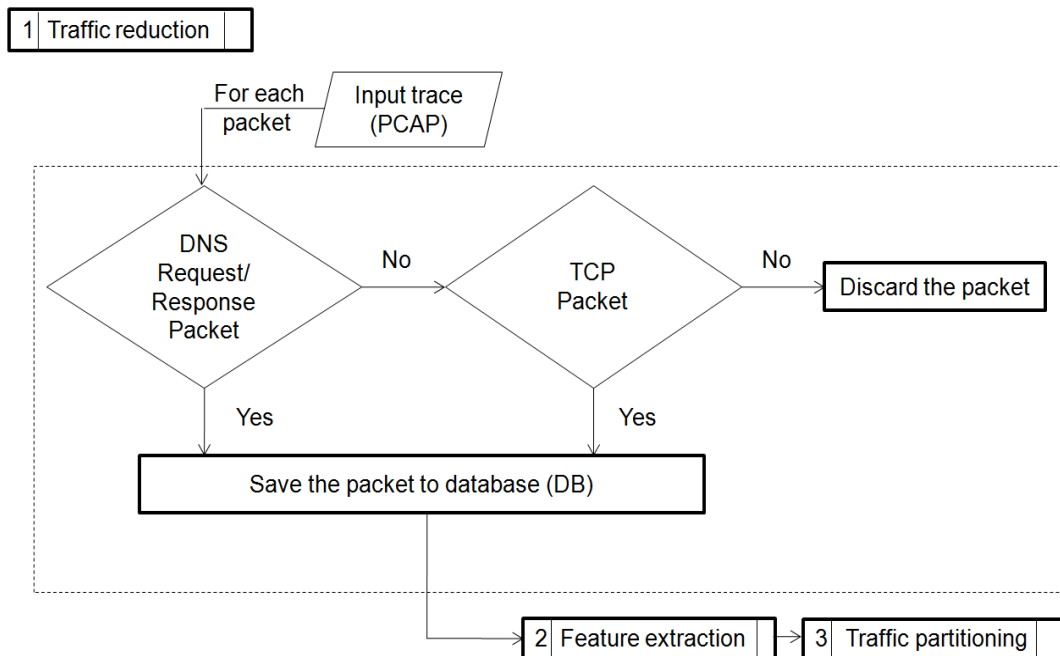


Figure 3. The procedure in the traffic reduction stage.

3.4 Feature extraction

Since bots always contain specific behaviors that are different from normal users' behaviors, we can detect bots by extracted features from bot behaviors.

3.4.1 Feature extraction from DNS packets

We observed that bots send DNS queries periodically in a time period since bot behaviors always involve DNS queries. For example, Figure 4 shows the distribution of botnet DNS query packets. We may use some features that we observed from bots specific behaviors to detect bots, such as total number of DNS packets between query and response packets, total times that a node used the same IP addresses, etc. Those packets that are related to bot DNS features will be stored in the database.

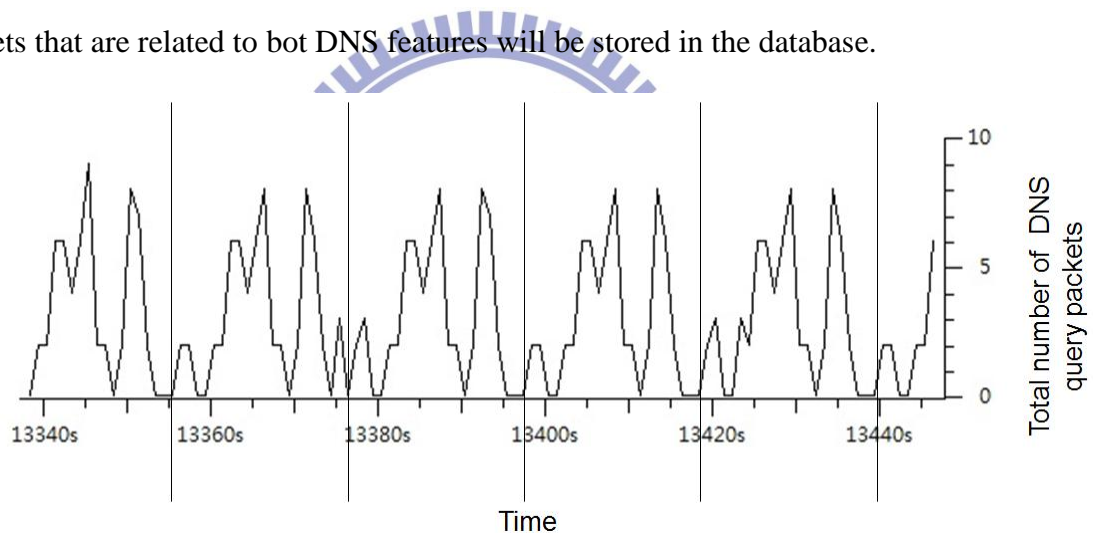


Figure 4. The distribution of botnet DNS query packets. [x axis: seconds; y axis: number of DNS query packets].

3.4.2 Feature extraction from TCP connection packets

Figure 5 shows the distribution of botnet TCP request packets. It illustrates that the TCP request packets distribution of botnet traffic is periodical. We use some features that we observed from bots specific behaviors to detect bots, such as packets count per second, bytes count per packet, etc. Those packets that are related to bot TCP features will be stored in the database.

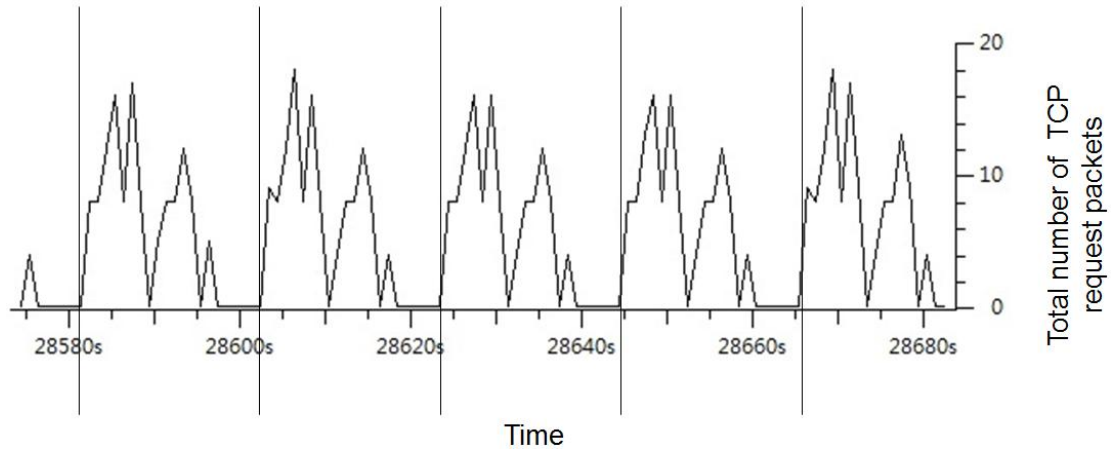


Figure 5. The distribution of botnet TCP request packets. [x axis: seconds; y axis: number of TCP request packets].

3.5 Fuzzy pattern recognition for DNS and TCP phases

As mentioned before, bot behaviors can be obfuscated to evade a string signature-based detection system. Therefore, we propose a behavior-based botnet detection algorithm in cloud computing environments (BBDC) that uses fuzzy pattern recognition to identify bots, including obfuscated bots. In the proposed BBDC algorithm, the fuzzy pattern recognition has two phases: DNS phase and TCP phase. In the DNS phase, we determine if it is a bot based on DNS features that were collected from the feature extraction stage, as shown in Figure 6. If it is not a bot, then the input trace is passed to the TCP phase. Otherwise, the input trace is identified as a bot. In the TCP phase, we detect a bot based on TCP features that were collected from the feature extraction stage, as shown in Figure 7. Both phases use fuzzy pattern recognition to classify bot and non-bot behaviors based on the max membership principle. Each membership function corresponds to a state and has a membership value, which will be described in the DNS and TCP phases later. The BBDC algorithm will find a max membership value, and the bot trace or normal trace is in

the state associated with this max membership value.

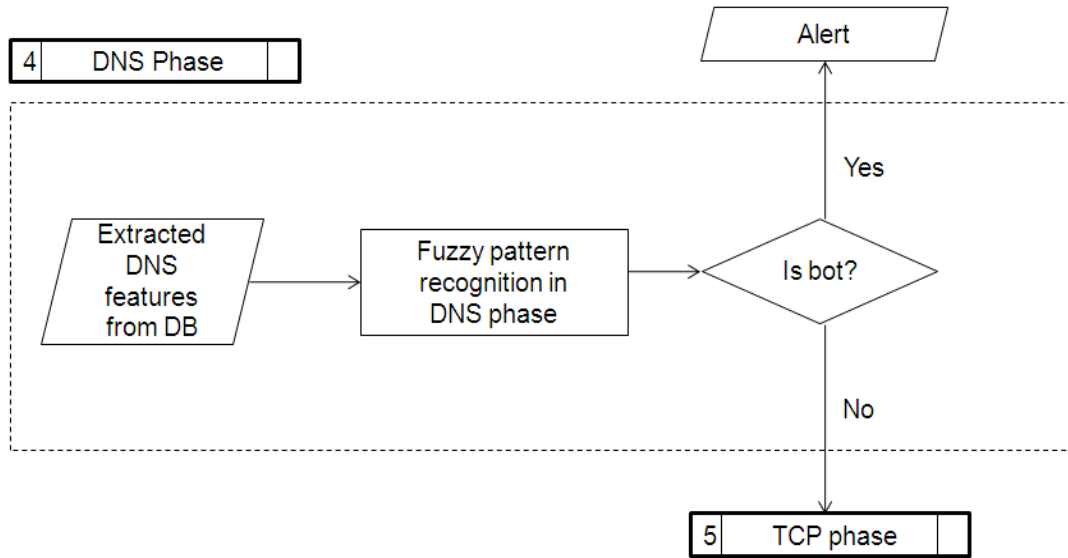


Figure 6. The DNS phase of the proposed botnet detection algorithm.

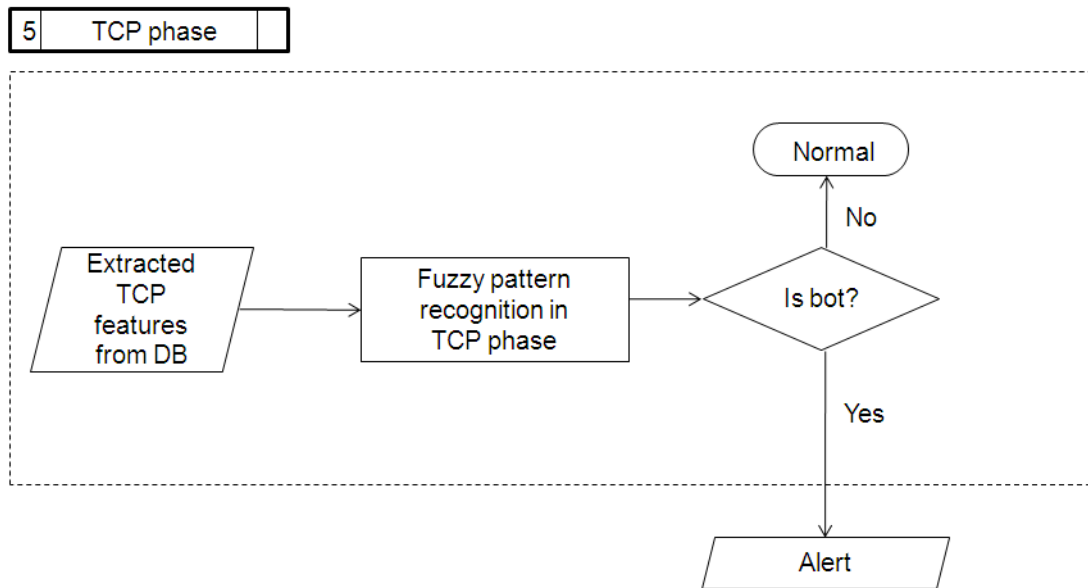


Figure 7. The TCP phase of the proposed botnet detection algorithm.

3.5.1 DNS phase

In the DNS phase, we define a packet features vector $x = (\alpha, \beta, \gamma, \lambda)$. α is a set of time intervals $\{\alpha_i \mid 1 \leq i \leq n\}$ between DNS query and response packets, where α_i is the length of time interval i between DNS query and

response packets, and n is the total number of DNS queries; β is a set of total number of DNS query packets $\{\beta_i | 1 \leq i \leq n\}$ in contrast to α , where β_i is the total number of DNS query packets in contrast to α_i , and n is the total number of DNS queries; γ is a set of $\{\gamma_i | 1 \leq i \leq N\}$, where γ_i is total times of the i^{th} IP address used by this node, and N is the number of IP addresses that a node used. λ is total number of DNS query and response packets per second. Figure 8 shows the fuzzy pattern recognition with max membership principle in the DNS phase. In this phase, we define four states and their associated membership functions, as described in the following.

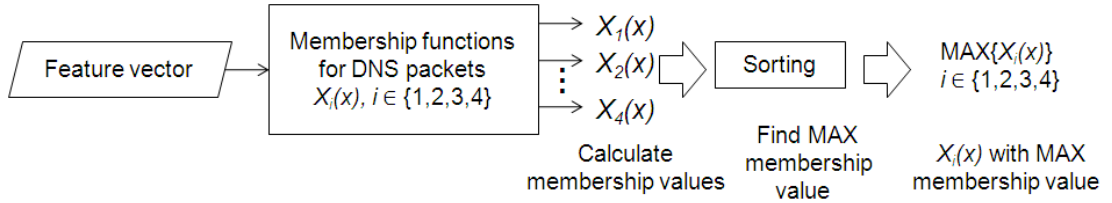


Figure 8. Fuzzy max membership principle in DNS phase.

- *Bot trace states*

(a) *Normalized abnormal variance of total number of DNS packets between query and response packets*

An active malicious DNS traffic usually has a large packet count in a time period. More DNS packets in a time period lead to a higher membership value. We define a membership function X_1 for calculating the normalized abnormal variance of total number of DNS packets between query and response packets, as follows:

$$X_1(x) = \begin{cases} \frac{\text{MAX}(\sum_{j=1}^N (\beta_{\alpha_{i,j}} - \bar{\beta}_{\alpha_i})^2)}{\sum_{i=1}^n (\sum_{j=1}^N (\beta_{\alpha_{i,j}} - \bar{\beta}_{\alpha_i})^2)}, & \sum_{j=1}^N (\beta_{\alpha_{i,j}} - \bar{\beta}_{\alpha_i})^2 > T_{x_1} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

for $i \in \{1, 2, \dots, n\}$, where N is the duration of time interval α_i in seconds, n is the total number of DNS queries, $\beta_{\alpha_{i,j}}$ is the total number of DNS

query packets in time interval α_i at the j^{th} second, and T_{x_1} is the threshold of being abnormal variance of DNS packets.

(b) Normalized abnormal total times that a node used the same IP addresses

Bots may contact specific IP addresses many times in their execution period. Therefore, we calculate the contact times per IP address to identify abnormal connections by bots. We define a membership function X_2 for calculating normalized abnormal total times that a node used the same IP addresses.

$$X_2(x) = \begin{cases} \frac{\text{MAX}(\gamma_i)}{\sum_{i=1}^N \gamma_i}, & \gamma_i > T_{x_2} \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i \in \{1, 2, \dots, N\} \quad (2)$$

where N is the number of IP addresses that a node used, γ_i is total times of the i^{th} IP address used by this node, and T_{x_2} is the threshold of the abnormal contact times per IP address.

(c) Normalized abnormal total number of DNS query and response packets per second

Bots may send DNS query packets many times in their execution time. Therefore, we calculate the times of DNS queries per second to identify abnormal DNS queries. We define a membership function X_3 for calculating normalized abnormal total number of DNS query and response packets per second.

$$X_3(x) = \begin{cases} \frac{\text{MAX}(\lambda_i)}{\sum_{i=1}^N \lambda_i}, & \lambda_i > T_{x_3} \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i \in \{1, 2, \dots, N\} \quad (3)$$

where N is the duration of an input trace in seconds, λ_i is the total number of DNS query and response packets in the i^{th} second, and T_{x_3} is the threshold of the total number of DNS query and response packets per

second.

- *Normal trace state*

We define a membership function X_4 for calculating the probability of being a normal trace.

$$X_4(x) = 1 - \text{MAX}\{X_1(x), X_2(x), X_3(x)\} \quad (4)$$

3.5.2 TCP phase

In the TCP phase, we define a packet features vector $x = (\alpha, \beta, \gamma, \lambda)$. α is a set of time intervals $\{\alpha_i | 1 \leq i \leq n\}$ between TCP request and response packets, where α_i is the length of time interval i between TCP request and response packets, and n is the total number of TCP request packets; β is a set of total number of TCP packets $\{\beta_i | 1 \leq i \leq n\}$ in contrast to α , where β_i is the total number of TCP packets in contrast to α_i , and n is the total number of TCP requests; γ is a set of total number of bytes $\{\gamma_i | 1 \leq i \leq n\}$ in contrast to α , where γ_i is the total number of bytes in contrast to α_i , and n is the total number of TCP requests. λ is total number of TCP request and response packets per second. Figure 9 shows the fuzzy pattern recognition with max membership principle in the TCP phase. In this phase, we define six states and their associated membership functions, as follows:

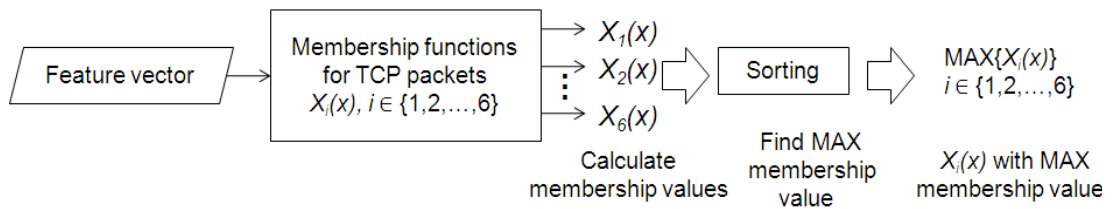


Figure 9. Fuzzy max membership principle in TCP phase.

- *Bot trace states*

(a) *Normalized abnormal packets count per second*

If a TCP connection sent too many requests in a second, the TCP packets count per second would reflect the abnormal behavior. We define a membership function X_1 for calculating the normalized abnormal packets count per second.

$$X_1(x) = \begin{cases} \frac{\beta_t/\alpha_t}{T_{x_1}} - 1, & 1 < \frac{\beta_t/\alpha_t}{T_{x_1}} < 2 \\ 1, & \frac{\beta_t/\alpha_t}{T_{x_1}} > 2 \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where β_t is the total number of TCP packets in an input trace, α_t is the duration of an input trace in seconds, and T_{x_1} is the threshold for abnormal packets count per second.

(b) *Normalized abnormal bytes count per packet*

If a bot master wants to send commands to other bots, the bytes count per TCP packet will reflect the abnormal behavior. We define a membership function X_2 for calculating the normalized abnormal bytes count per packet.

$$X_2(x) = \begin{cases} \frac{\gamma_t/\beta_t}{T_{x_2}} - 1, & 1 < \frac{\gamma_t/\beta_t}{T_{x_2}} < 2 \\ 1, & \frac{\gamma_t/\beta_t}{T_{x_2}} > 2 \\ 0, & \text{otherwise} \end{cases} \quad (6)$$

where γ_t is the total number of bytes in an input trace, β_t is the total number of TCP packets in an input trace, and T_{x_2} is the threshold for abnormal bytes count per packet.

(c) *Normalized abnormal variance of total number of TCP packets between request and response packets*

An active malicious TCP traffic usually has a large packet count in a time period. More TCP packets in a time period lead to a higher membership

value. We define a membership function X_3 for calculating the normalized abnormal variance of total number of TCP packets.

$$X_3(x) = \begin{cases} \frac{\text{MAX}(\sum_{j=1}^N(\beta_{\alpha_{i,j}} - \bar{\beta}_{\alpha_i})^2)}{\sum_{i=1}^n(\sum_{j=1}^N(\beta_{\alpha_{i,j}} - \bar{\beta}_{\alpha_i})^2)}, & \sum_{j=1}^N(\beta_{\alpha_{i,j}} - \bar{\beta}_{\alpha_i})^2 > T_{x_3} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

for $i \in \{1, 2, \dots, n\}$, where N is the duration of time interval α_i in seconds, n is the total number of TCP request packets and $\beta_{\alpha_{i,j}}$ is total number of TCP request packets in time interval α_i at the j^{th} second, and T_{x_3} is the threshold of the variance of total number of TCP packets.

(d) Normalized abnormal variance of total number of bytes

An active malicious TCP traffic usually has a large byte count of TCP packets in a time period. More bytes in a time period lead to a higher membership value. We define a membership function X_4 for calculating the normalized abnormal variance of total number of bytes.

$$X_4(x) = \begin{cases} \frac{\text{MAX}(\sum_{j=1}^N(\gamma_{\alpha_{i,j}} - \bar{\gamma}_{\alpha_i})^2)}{\sum_{i=1}^n(\sum_{j=1}^N(\gamma_{\alpha_{i,j}} - \bar{\gamma}_{\alpha_i})^2)}, & \sum_{j=1}^N(\gamma_{\alpha_{i,j}} - \bar{\gamma}_{\alpha_i})^2 > T_{x_4} \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

for $i \in \{1, 2, \dots, n\}$, where N is the duration of time interval α_i in seconds, n is the total number of TCP request packets and $\gamma_{\alpha_{i,j}}$ is the total number of bytes in time interval α_i at the j^{th} second, and T_{x_4} is the threshold of the variance of abnormal total number of bytes.

(e) Normalized abnormal total number of TCP request and response packets per second

Bots may send TCP request and response packets many times in their execution periods. Therefore, we calculate the total number of TCP request and response packets per second to identify abnormal behaviors. We define a membership function X_5 for calculating the normalized abnormal total number of TCP request and response packets per second.

$$X_5(x) = \begin{cases} \frac{\text{MAX}(\lambda_i)}{\sum_{i=1}^N \lambda_i}, & \lambda_i > T_{x_5} \quad \text{for } i \in \{1, 2, \dots, N\} \\ 0, & \text{otherwise} \end{cases}$$

where N is the duration of an input trace in seconds, λ_i is the total number of TCP request and response packets in the i^{th} second, and T_{x_5} is the threshold of the total number of TCP packets per second.

- *Normal trace state*

We define a membership function X_6 for calculating the probability of being a normal trace.

$$X_6(x) = 1 - \text{MAX}\{X_1(x), X_2(x), X_3(x), X_4(x), X_5(x)\} \quad (10)$$

3.5.3 Observation

We observed 50 bots and 50 normal traces. Each trace lasts for two hours. In these experiments, we found that there are different behaviors between bots and normal traces. Figure 10 shows some observations of six bot characteristics. In Figure 10(a) and 10(b) show the statistics about the variance of packet inter arrival time and the variance of bytes per packet, respectively. In Figure 10(c) and 10(d) show the statistics about the number of packets between request and response packets and the contact times per IP address. In Figure 10(e) and 10(f) show the statistics about the average number of packets per second and the average number of bytes per packet.

Each characteristic reflects different behaviors of some bots. To reduce the false-positive rate, we choose more bot's behavior characteristics that can characterize botnets. To reflect the behaviors of IRC bots, the contact times that a node used the same IP addresses and the total number of DNS query and response packets per second can be used to detect IRC bots. To

reflect the behaviors of HTTP bots, the packet count per second and the byte count per packet can be used to detect HTTP bots.



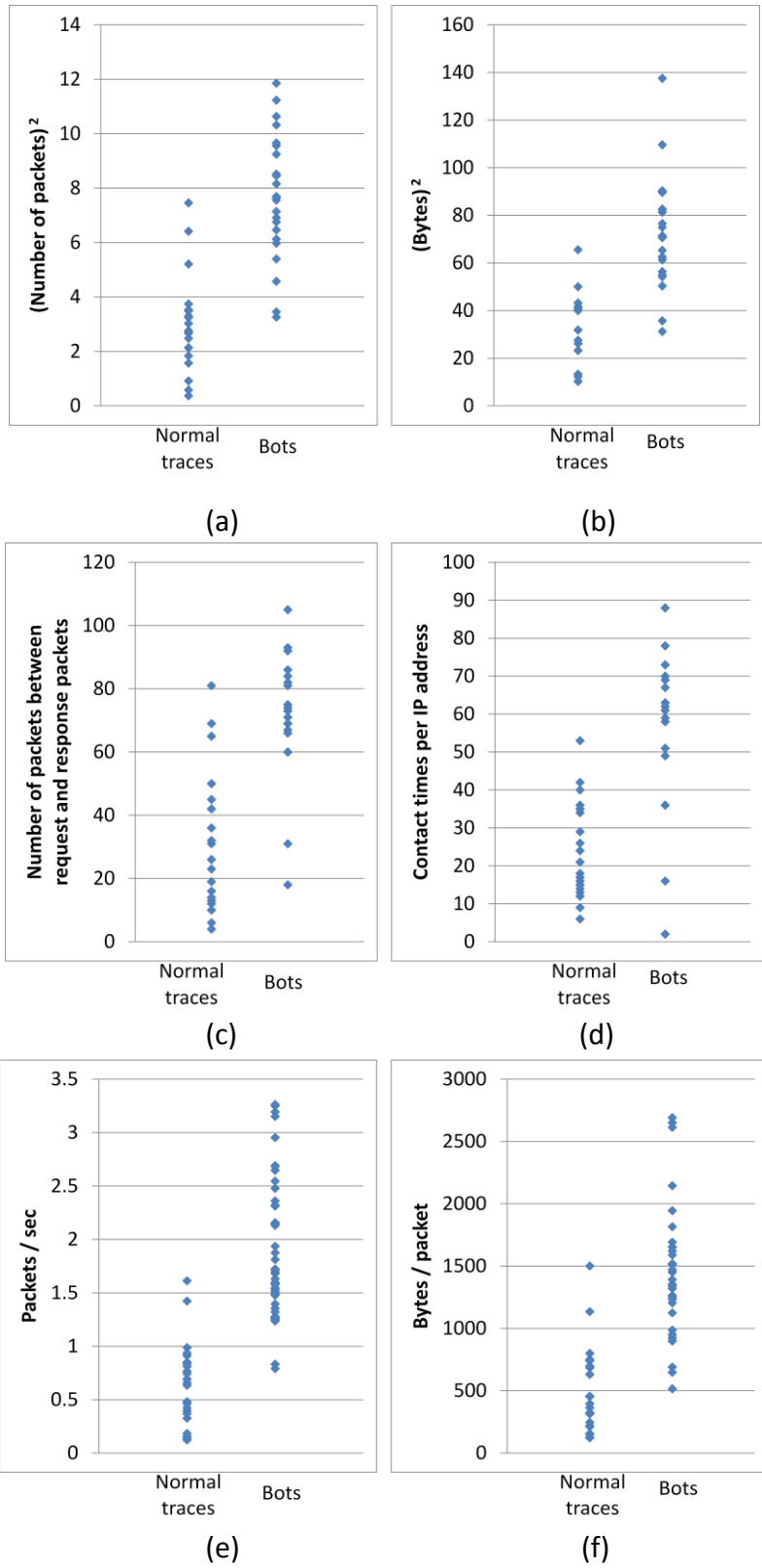


Figure 10. Some observations of six bot characteristics.

Chapter 4

Performance Evaluation

4.1 Traces collection

To collect real botnet traces, we installed an unpatched Windows XP SP3 in a virtual machine using Ubuntu and Virtualbox, and executed 250 real bot samples inside a Windows environment (HoneyTrap). We used a share folder to manage honeytraps (1 to N), as shown in Figure 10. Both input and output network traffic of the virtual machine were recorded by a Recorder and stored in an Apache database via DBInserter. Among 250 bots, only 240 bots had network traces. The rest of bots were not executable. The botnet traces were recorded for 2 hours via Recorder. Both the packet header and complete packet payload were stored in the database for further botnet analysis.

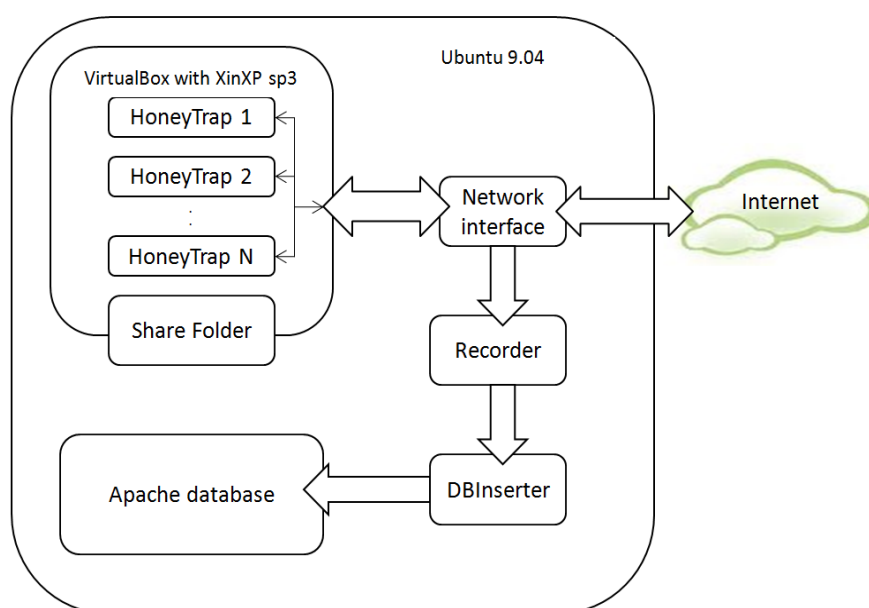


Figure 10. Experimental environment for botnet traces collection.

4.2 Test results of botnet traces

We used real botnet traces to evaluate our BBDC algorithm. Statistics of the botnet traces and the false negative rate are shown in Table 1. The evaluation result shows that BBDC a low false negative rate (FNR), 4.17%.

Table 1. Botnet traces statistics and false negative rate.

Number of bots	Inactive bots	Active bots	Traffic reduction rate	Number of active bots detected	False negative rate
250	10	240	75.4%	230	4.17%

We also collected two normal traces (T1 through T2) to evaluate the BBDC's traffic reduction rate and the false positive rate (FPR). T1 traces were collected from the National Chiao Tung University's campus beta site. We collected 695 traces in T1 for 2 hours. T2 traces were obtained from our MAL laboratory. We collected 5 traces in T2 from laboratory's members. These traces contain various types of benign applications using IRC, HTTP, and P2P, etc. In Table 2, we found that the BBDC achieves high reduction rates and low FPRs.

Table 2. Normal traces statistics and false positive rates.

Test site	Number of normal traces	Traffic reduction rate	Number of traces identified as bots	False positive rate (FPR)
T1	695	77.3%	24	3.453%
T2	5	73.1%	0	0%

Figure 11 shows the proposed botnet detection system's total execution time in Windows azure cloud and host-based environment. We used 1 to 5 server instances

(SIs) to evaluate the total execution time in our proposed system. In the experiments, we evaluate both host-based and cloud-based execution time. Experimental results show that the total execution time can be reduced in proportion to number of SIs used. The cloud-based system is 4.73 times faster than the host-based system in terms of total execution time with 5 SIs used.

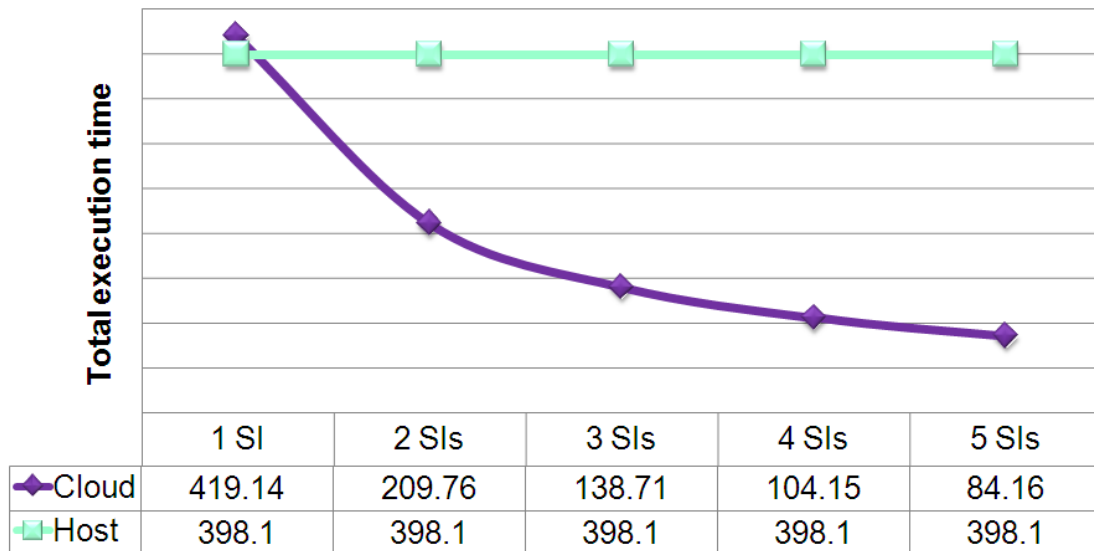


Figure 11. Total execution time (sec) for 950 traces using various number of server instances (SIs) in Windows Azure cloud.

Figure 12 shows the false positive bots distribution among bot behaviors in the DNS phase. Figure 13 shows the false positive bots distribution among bot behaviors in the TCP phase. Since our proposed botnet detection system may have false positive bots, we want to find out the causes of such bot features. Both figures show the bot features that cause normal traces being identified as bots. By these statistics, we can adjust membership functions to reduce the proposed system's FPR in the future.

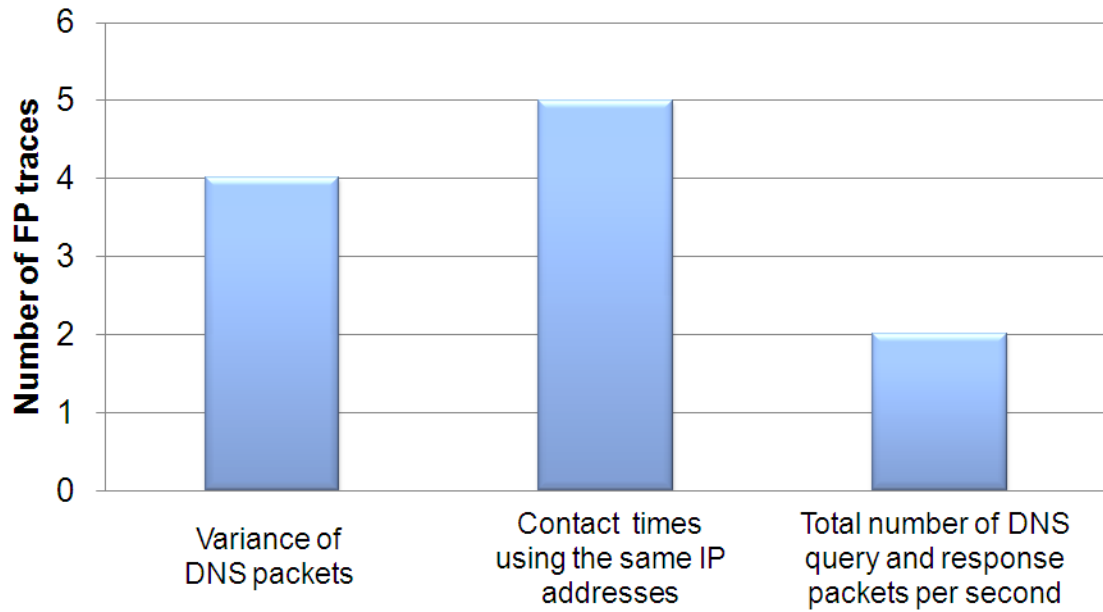


Figure 12. False positive bots distribution with respect to bot features in DNS

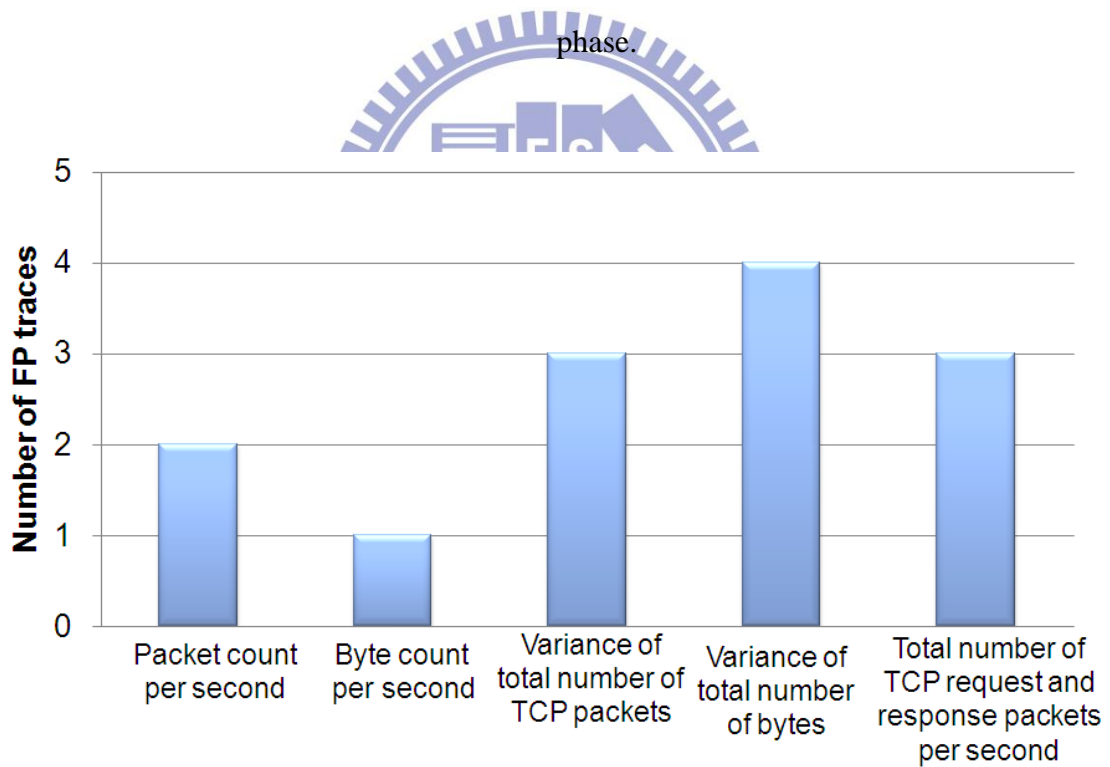


Figure 13. False positive bots distribution with respect to bot features in TCP phase.

Figure 14 shows the false negative bots distribution for each bot type, IRC or HTTP bots. Since bots may evade and obfuscate the botnet detection system, our proposed system may have false negative bots. By observing the bots' traces which cause false negative (FN), we may modify or add bot features to reduce FN bots.

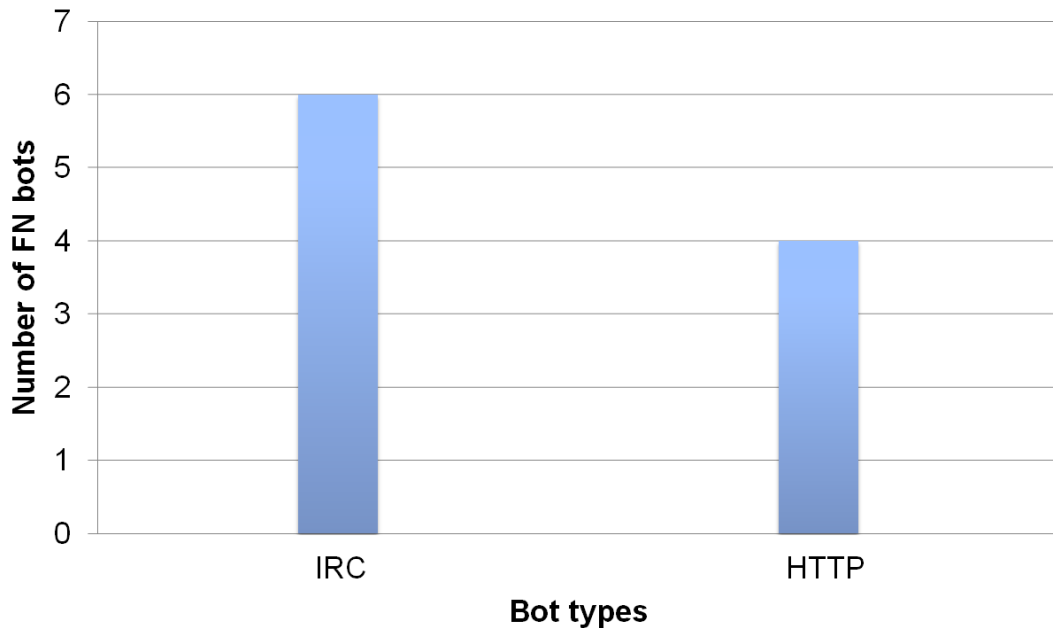


Figure 14. False negative bots distribution for different bot types.

We also evaluate the execution time in each stage in our botnet detection system. The traffic reduction phase spent 10% of time. The DNS phase spent about 37% of time. The TCP phase spent about 53% of time. Since the DNS and TCP phases spent most total execution time, we port the proposed system into a cloud environment in order to reduce the botnet detection system's total execution time.

In Table 3, we compare the proposed BBDC and the other three existing botnet detection methods. Unlike these three existing methods, BBDC used 250 real bot samples to emulate real botnet traffic. Except Yu [5], the proposed BBDC performs better than Park [4] and Lin [1] in terms of false positive rate and false negative rate. Note that the botnet traces of Yu [5] only contain four fabricated bots, not real bots.

Table 3. Comparison of different behavior-based botnet detection methods.

Approach	BBDC (proposed)	Park [4]	Yu [5]	Lin [1]
Basic idea	Using fuzzy pattern recognition cloud to detect bots	Using static analysis with a data-mining technique to detect bots	Using feature streams to match whole input trace to detect bots	Using fuzzy pattern recognition filtering algorithm to detect bots
Botnet trace	Real botnet traffic	Real botnet traffic	4 fabricated bots	Real botnet traffic
True positive rate	95.83%	94.35%	100%	90.41%
False positive rate	3.429%	4.393%	14.7%	9.59%
False negative rate	4.17%	5.65%	0%	5.41%
Bot samples	250	110	4	250
Categories of bot samples	IRC+HTTP	IRC+HTTP	IRC	IRC+HTTP
Total execution time (host-based)	398.1 sec.	N/A	N/A	N/A
Total execution time (cloud-based with 5-server instances)	84.2 sec.	N/A	N/A	N/A

Chapter 5

Conclusion

5.1 Concluding remarks

In this thesis, we have presented a behavior-based botnet detection algorithm using a cloud computing environment to speed up botnet detection. Our algorithm can reduce false positive and false negative rates compared with other behavior-based detection systems. We reduce the total execution time by traffic reduction and cloud computing. Since the proposed algorithm identifies different protocols used in an input trace and detect bots based on bot behaviors in each protocol, it can have a high detection rate (true positive rate). We use a fuzzy pattern recognition with max-membership principle to perform bot behaviors matching in the DNS and TCP phases. We have used real bots to generate real botnet traces for evaluating the proposed BBDC algorithm. As a result, the proposed botnet detection system can be feasible in a real world. Experimental results show that our proposed system can achieve a high true positive rate of 95.83% and a low false positive rate of 3.429% in the Windows Azure cloud computing platform. Furthermore, the proposed cloud-based system with five server instances is 4.73 times faster than a host-based system.

5.2 Future work

In the future, we will extend our work to detect P2P botnets by including P2P bots features in the proposed botnet detection system. Moreover, we want to reduce the FPR and FNR of the proposed botnet detection system by extracting more bot's behavior features that can characterize botnets.

References

- [1] K. Wang, C. Huang, S. Lin, Y. Lin, "A fuzzy pattern-based filtering algorithm for botnet detection," *Computer Networks*, 2011 (accepted for publication).
- [2] "Snort," [Online]. Available: <http://www.snort.org/>.
- [3] B. AsSadhan, José M. F. Moura and D. Lapsley, "Periodic behavior in botnet command and control channels traffic," in *Proc. IEEE Global Telecommunication Conf.*, 2009, pp. 2157-2162.
- [4] Y. Park, Q. Zhang, D. Reeves, V. Mulukutla, "AntiBot: clustering common semantic patterns for bot detection," in *Proc. IEEE Computer Software and Applications Conf.*, 2010.
- [5] X. Yu, X. Dong, G. Yu, Y. Qin, D. Yue, "Online botnet detection based on incremental discrete Fourier transform," *Networks Journal*, May 2010, pp. 568-576.
- [6] J. Kang, J. Zhang, Q. Li, Z. Li, "Detecting new P2P botnet with multi-chart CUSUM," in *Proc. IEEE Wireless Communications and Trusted Computing Conf.*, 2009.
- [7] W. Lu, M. Tavallaei, G. Rammidi, A.A. Ghorbani, "BotCop: An online botnet traffic classifier," in *Proc. IEEE Communication Networks and Services Research Conf.*, 2009.
- [8] F. Alserhani, M. Akhlaq, I.U. Awan, A.J. Cullen, "Detection of coordinated attacks using alert correlation model," in *Proc. IEEE Progress in Informatics and Computing Conf.*, 2010.
- [9] M. Szymczyk, "Detecting botnets in computer networks using multi-agent technology," in *Proc. IEEE Dependability of Computer Systems Conf.*, 2009.
- [10] L. Braun, G. Munz, G. Carle, "Packet sampling for worm and botnet detection in

TCP connections”, in *Proc. Network Operations and Management Symposium (NOMS)*, 2010.

[11] “Windows azure,” [Online]. Available:

<http://msdn.microsoft.com/zh-tw/windowsazure/cc947856>.

[12] P. Sinha, A. Boukhtouta, V.H. Belarde, M. Debbabi, “Insights from the analysis of the Mariposa botnet”, in *Proc. IEEE Risks and Security of Internet and Systems Conf.*, 2010.

[13] “Social engineering,” [Online]. Available:

http://en.wikipedia.org/wiki/Social_engineering_%28security%29

