國立交通大學

資訊科學與工程研究所

碩 士 論 文

基於相似關聯性探勘之異常網域偵測

Abnormal Domain Detection by Discovering Mutual Associations from Large-Scale Traffic Logs

研 究 生：王琮瑋

指導教授：彭文志　教授

中 華 民 國　九 十 八 年 十一 月

基於相似關聯性探勘之異常網域偵測
Abnormal Domain Detection by Discovering Mutual Associations from Large-Scale Traffic Logs

研 究 生：王琮瑋　　Student：Tsung-Wei Wang

指導教授：彭文志　　Advisor：Wen-Chih Peng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

November 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年十一月

基於相似關聯性探勘之異常網域偵測

學生：王琮瑋　　指導教授：彭文志

國立交通大學資訊科學與工程研究所碩士班

摘　　要

安全性是網路服務中的一個主要關注。因此，提供網路服務可靠的機制來確保用戶的安全已成為一個重要問題。在網路安全的領域中，殭屍網路是其中特別重要的一個議題，攻擊者可以透過控制大量的個人電腦，來進行許多惡意的網路威脅。一般來說，控制殭屍網路包括兩個機制：傳播策略和通信管道的建立。最早期建立溝通管道的方法是透過固定的 IP。從 2007 年開始，fast flux 的技術被應用在 Storm Worm 殭屍網路中建立一種新的溝通管道，這使得殭屍網路與其擁有者之間的溝通管道更難以被停止。在 2008 年年底，Conficker.C 進一步改善了溝通管道，利用隨機網域生成的演算法來改善及延長溝通管道的建立。由殭屍網路的演化中可以發現，一種可以用於偵測快速變化的網域及 IP 地址的異常網域偵測可以更快速的找出溝通管道並加以阻擋。在這項研究中，我們使用 IP 位址之間的關連性來作為異常網域偵測的基礎。具體來說，我們希望利用單一筆 DNS 記錄中 IP 地址的組合來偵測網域的異常。根據我們的觀察，正常的 DNS 記錄中，IP 間的關連性會比異常網域的關連性來的高，這些異常網域包括 fast flux 或是隨機產生的網域。在這個觀察基礎上，我們設計了異常網域偵測的 framework。這個 framework 包含了四大部分，首先我們由網路流量中建立出網域與 IP 位址間的關連性，接著我們利用這些關連性來排除記錄中由於 proxy 所造成的 noise，在第三個步驟我們進一步估算出 IP 位址彼此間的關連性，最後再利用這些關連性資訊找出異常的網域。

Abnormal Domain Detection by Discovering Mutual Associations from Large-Scale Traffic Logs

student：Tsung-Wei Wang　　Advisors：Dr. Wen-Chih Peng

Department of Computer Science
National Chiao Tung University

ABSTRACT

Security is one of the major concerns for web services. In particular, botnet which control a group of computers for malicious purpose brings a lot of online threats. Generally speaking, controlling a botnet includes two mechanisms: propagation strategies and communication channel establishment. Classic communication channels are established via hardcoded IP. Recently, fast-flux hosting characterized rapid changes of IP address, is widely applied in Storm Worm Botnet. More recently, Conficker.C f takes fast flux to the next level by using dynamically generated domain names to find out its control servers and load instructions from them. The history of advances of communication strategy suggests that a domain detection model which enables to handle not only rapid changes of IP address but also that of domain names is desired.

In this study, we propose to highlight the utility of mutual associations to offer anomaly domain detection. Specifically, our framework constructs a detection model based on mutual associations among IP classes, which results in a set of profiles. Afterwards, the predicted mutual associations are utilized to estimate the degree of abnormality for a target domain, which set us apart from current approaches.

誌　　謝

研究所兩年多的時間很快就過去了，過程中碰到了許多的挫折與困難，謝謝彭文志老師這幾年來的指導與包容，也很感謝實驗室學長們的照顧與支持，不管生活或是學習上碰到什麼困難，學長們總是可以提供給我很好的建議，跟大家的相處與討論讓我學習到了許多，在兩年多的過程當中，從大家身上發現許多值得學習的地方，也發現了許多自己不足之處。

也要感謝我家人，始終在我的身旁給我許多的鼓勵與支持，讓我可以更專注在研究學習上，很謝謝你們為我所作的一切。

最後要特別感謝 young 的支持與陪伴，在我最感到孤單無助的時候，給我許多的安慰與幫忙，讓我可以撐過碰到的難關。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advance of internet technology not only improves the quality of our lives but also raises the issue of cybercrimes. Fast-flux hosting supports a variety of cybercrime activities. One variant of fast-flux hosting uses rapid change of IP addresses to hide their location of website where illegal activities are performed. In particular, malicious use of fast-flux takes the form of hiding the command and control server using compromised machines (”zombies”) that are used in DDoS, spam campaigns, phishing attacks, malware delivery by botnet operators [1][2]. Moreover, advanced computer worm such as Conficker is controlled by a pseudo-random domain name generator to hide the actual location of control server [3]. Some of users fall for these attacks by performing clicks on malicious websites, which can lead to exposure of credentials information, downloading malware, or other undesirable effects. As a result, the development of effective and efficient abnormal domain detection framework is highly desired.

The most related work about abnormal domain detection is fast-flux service network (FFSN) detection. Previous approaches for FFSN detection focus on modeling the characteristic of rapid IP changing as a feature to distinguish fast-flux domains from benign domains. The authors in [4] train a classifier according to designed features such as number of unique A records returned in all DNS lookups to decide whether a given domain is FFSN or not. While they provide a formal detection model for FFSN detection, it requires to monitor traffic logs for a period of time to obtain such feature values. In order to detect FFSN as early as possible from daily large-scale traffic logs, it is necessary to establish a reliable framework from different perspective.

Internet service providers (ISP) that offer their customers access to the internet will log customers' connection events over time. Traffic stream provides valuable information. An effective data mining mechanism to summarize useful information from raw traffic stream is hence an important issue. For example, a useful information for entities (e.g., website) which includes website chains can assist security expert to track malicious hosts or identify victims effectively. Similarly, a

profile of trustworthiness record of a website can serve as a reference for filtering websites of low-quality. A profile of static traffic statistics such as number of visited websites over time can assist security expert to notice abnormal behaviors. There is no guarantee that which feature leads to the best detection results. As a result, an entity profile including multiple types of features can provide a more comprehensive summary and is expected to be more effective during abnormal domain detection.

An automatic abnormal domain detection mechanism is also desired to block suspicious websites as early as possible. As reported from [2], fast flux hosting supports a variety of cybercrime activities such as fraud, identity theft, online scams and is considered one of the most serious threats today. Adversaries usually host or distribute malicious content via FFSN to better operate huge amount of zombie hosts to postpone being blocked by ISP or antivirus companies. Newly invented computer worm such as Conficker goes to another level by performing a pseudo random domain name generator to last the communication channel between control node and zombies [5]. Therefore, designing an automatic and efficient domain detection framework is necessary to reduce the cybercrime activities.

In this study, we develop a profile-based framework for detecting abnormal domains, which scales to large traffic logs and enables for handle noise in traffic logs. The framework consists of four components: 1) preprocessing, 2) proxy filtering, 3) mutual association discovery, and 4) abnormal domain estimation. Specifically, we first transform raw traffic logs into an adjacent matrix. Then, based on our observation on proxy servers, we design a metric to estimate proxy scores for an IP class. Third, to exploiting the relations between domains and IP classes, we design an algorithm to measure the mutual association between a pair of IP classes. Finally, regarding a DNS record, we investigate the connectivity of the domain and estimate its anomaly score based on computed mutual associations.

Instead of detecting abnormal domains by tracking and modeling rapid IP changing behavior over time as [4], we propose to detect abnormal domain by discovering mutual associations between website and IP addresses in a single time slot. In essence, the task of performing repeated DNS lookups for a period of time to derive the number of unique A records for FFSN detection is time-consuming. In contrast, discovering mutual associations from website and IP address is expected to be practical. Consequently, we focus on designing an automatic framework for abnormal domain detection by mutual association discovery. The discovered mutual associations can be utilized for anomaly score computation. In addition, to find accurate mutual associations, it is required to eliminate noises (e.g., proxy) from raw traffic logs, which lead us to achieve another contribution of this study.

In summary, the main contribution of our study is as follows:

- The development of approach for discovering mutual associations.
- The design of parameter free framework for abnormal domain detection by exploiting mutual associations.
- The development of approach for proxy detection algorithm by exploiting co-occurrences among domains

The remainder of this paper is organized as follows. In Section 2, we briefly introduce related work about fast-flux service network detection. In Section 3, we present notations, data representation and then give formal problem definitions. In Section 4, we describe the proposed framework for abnormal domain detection framework in detail. In Section 5, we conduct several experiments, report quality studies and demonstrate the scalability of our framework. Finally, we offer our concluding remarks in Section 6.

Chapter 2

Related Works

In this section, we review the related work about fast-flux service network. Botnet (a.k.a. Zombie Network) is a serious issue in the field of network security. Many research work has been proposed to discuss and solve this problem[6][7]. Botnet can be caused by variant propagation strategies [8][9]. In addition to propagation strategies, there must be communication channels between the botnet owner and zombies. At early stage, botmaster uses a single rendezvous point to communicate with its zombies. However, the communication channel can be easily blocked by IP blacklist. As a result, fast flux technique was developed to foil IP blacklisting due to its rapid changes of IP address mapping to a domain name.

Fast flux is one of the issue caused by botnet. Adversaries usually host or distribute malicious content via botnet. Delivery malicious content by a single zombie host can be easily stopped by ISP or antivirus companies. As a result, a mechanism FFSN to better operate huge amount of zombie hosts is developed to postpone the time of being blocked by ISP or antivirus companies. Fast flux service network uses rapid and repeated changes to host and/or name server resource records, which result in two characteristics: 1) **IP address diversity**: IP address to which the domain name of an Internet host or name server resolves changes rapidly and 2) **No physical agent control**: the flux-agent that scammer can rely on has no guaranteed uptime [4][2][1].

The ICANN Security and Stability Advisory Committee (SSAC) recently release a report [2]. This Advisory describes the technical aspects of fast flux hosting and fast flux service networks. It explains how the DNS is exploited to encourage malicious activities that employ fast flux hosting, identifying the impacts of fast flux hosting, and calling particular attention to the way such attacks extend the malicious or profitable lifetime of the illegal activities conducted using these fast flux techniques. It describes current and possible methods of mitigating fast flux hosting at various points in the Internet. SSAC considers that with appropriate use of mitigation methods would make the practical mitigation methods universally available to registrants, ISPs, registrars and registries

(where applicable for each).

According to [4], fast flux hosting can be classified into three categories:

- Basic fast flux hosting where IP addresses of malicious web sites are fluxed
- Name Server (NS) fluxing where IP addresses of DNS name servers are fluxed
- Double flux, where IP addresses of web sites and name servers are fluxed.

The authors in [4] present the first empirical study of fast-flux service network($FFSNs$). They cast FFSNs detection as a classification problem which distinguishes FFSNs and other legitimate domains (e.g., Content Distribution Networks). Based on two characteristics of FFSNs mentioned previously, they train a classifier according to three designed features including 1) number of unique A records returned in all DNS lookups , 2) number of nameserver records in a single lookup, and 3) number of unique ASNs for all A records to decide whether a given domain is FFSN or not. The authors demonstrate the effectiveness of proposed features via a long-term monitoring (i.e., periodically performing DNS lookups to track measurements). Moreover, the authors provides three strategies to mitigate the threat posed by FFSNs by automatically collecting detected domains in a domain blacklist. In particular, the blacklist can be used 1) by an ISP to protect its clients from requests for fast-flux domains, 2) by an domain name register to shot down domains blacklist, and 3) by email servers to filter spam. While the authors developed effective empirical metrics for detecting new fast-flux domains automatically, we look forward to detect new fast-flux domains in an automatic and efficient way for large-scale traffic logs which set us apart from previous work.

Although fast-flux technique increases the difficulty to block the communication channel, new approach is proposed to against existing security solutions. Specifically, a computer worm, Conficker [3], discovered in early November 2008 further improves the way to control the botnet. So far, five variants of the Conficker worm are known. Recent estimates of the number of infected computers have been more notably difficult because of changes in the propagation and update strategy of recent variants of the worm [10]. In particular, Conficker takes 'fast-flux' to the next level by using dynamically generated domain names to find out its control servers and load instructions from them [5]. Current methods for Conficker detection use software technique to analyze the binary code of Conficker. Then, security experts can find out the full list of domains names generated by the pseudo-random generator inside Conficker[11]. However, to automatically deal with large scale traffic logs and to detect abnormal domains as early as possible is still a necessary and challenging task.

Chapter 3

Preliminary

In this section, we first introduce the data set and our data representation. Then, we give the formal problem definition.

3.1 Traffic Logs

The traffic log contains a set of events. Each event consists of a set of entities. In particular, we extract source IPs, destination domain names, hosted IPs, and timestamps. In our setting, we refer to a domain as an "domain" object, and refer to the IP class of a hosted IP as an "IP class" object respectively. We denote the set of domain object as H; we denote the set of IP class object as C. Each instance of two object types is refer to h and c respectively.

In this paper, we represent the collection of domain and IP classes as a bipartite graph. In particular, we consider each instance for each type of object as a node in the graph. The bipartite graph G is consisted by two set of vertices, V^H and V^C, and a set of hosting relations $E^{(H,C)}$. V^H corresponds to domain nodes; V^{PC} corresponds to IP class nodes. Furthermore, if a domain h hosts on an IP class c, then the hosting information is represented as $e(u, v)$ where u corresponds the domain node h and v refers to the IP class c. We use an adjacency matrix, $A_{m\times n}$, to model the relationship between the "domain" object and "IP class" object. Particularly, the subscripts m and n denotes the sizes of domain objects and IP class objects. Also, we represent the elements in a matrix at the i^{th} row and j^{th} column of the matrix a as $A(i, j)$. Each element $A(i, j)$ is determined by how many times a domain h_i hosted on IP class c_j.

3.2 Problem Formulation

Given a domain and its adopted IP classes in DNS record, how can we estimate the degree of abnormality for the domain and find the similarity for all its adopted IP classes? For example, if a domain adopts a combination of IP classes among which any pair of IP classes is seldom co-occurred, the domain could be a suspect domain, which are randomly generated for hiding their location on the internet.

To detect such domains, the following issues should be addressed. First, the detection framework should be scalable to large-scale traffic logs (daily traffic logs contain more than 200M records). Second, the detection framework should be able to handle noise data. Third, the detection framework should be automatic. To brief, the proposed solution for detecting abnormal domains should be scalable, noise insensitive, and parameter free.

3.3 Framework Overview

Our goal is to detect abnormal domains automatically as early as possible. Figure 3.1 illustrates the overview of our framework. Following describes formal statements for each component in our framework.

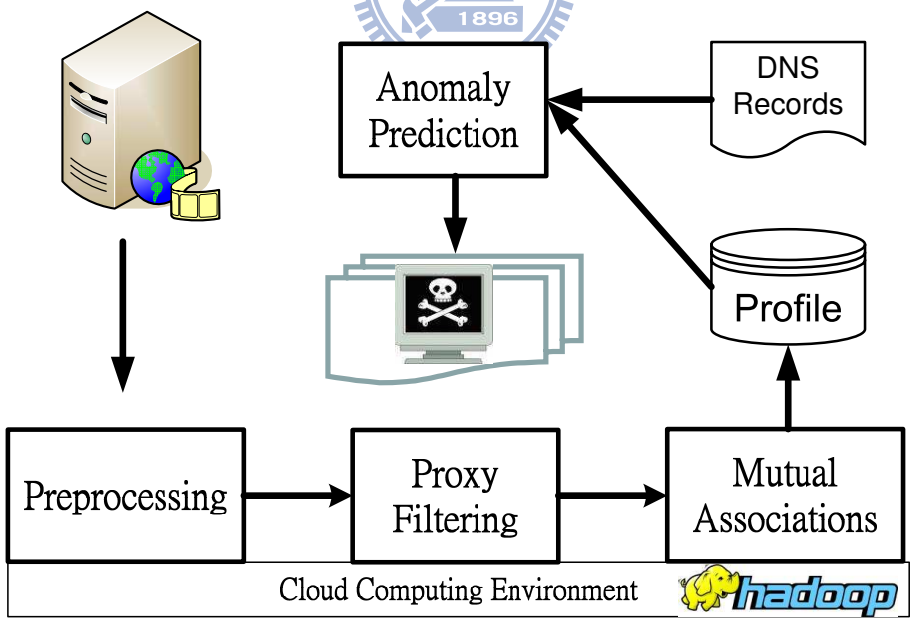


Figure 3.1: An overview of proposed framework for abnormal domain detection.

According to collected traffic logs, we learn that proxy servers cause major noise connections which will consequently mislead the following procedure for mutual association discovery. Hence, the first problem to abnormal domain detection is noise filtering which can be formally defined as follows:

- Problem 1: Noise Filtering

Given a set of collected domains, H, and a set of collected IP, C, **find** a filtering function g for IP classes in order to remove proxy servers from H.

Secondly, once we remove proxy-related connection records from collected traffic logs, we start to construct co-clustering information for each domain which can be formally defined as follows:

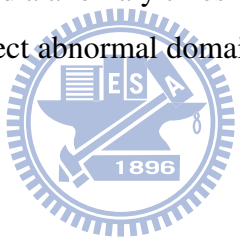
- Problem 2: Mutual Associations Discovery

Given a bipartite graph $G_{m \times n}$, **find** a function l to quantify mutual associations between any pair of IP class such that they shares similar adjacent domains.

After we feed mutual association information for each domain, we have to estimate the anomaly score for candidate domains based on co-clustering information so as to efficiently detect suspicious domains. The problem for abnormal domains detection is formally defined as follows:

- Problem 3: Abnormal Domain Detection

Given a mutual association matrix M and a anomaly threshold δ, **find** an anomaly scoring function f for measuring domains in order to detect abnormal domains whose anomaly score is no less than δ.



Chapter 4

A Framework to Detect Abnormal Domains

In this section, we will introduce each component in detail. Our framework mainly contains four procedures. First, the raw traffic logs are transformed into an adjacent matrix. Second, to mitigate the effect caused by noise entries, a proxy filtering algorithm is proposed to remove proxy-relevant entries from the adjacent matrix. After that, a mutual association algorithm is presented to build a mutual association matrix from filtered adjacent matrix so that the occurrence of each pair of IP classes can be modeled and quantified into profiles. Finally, the profiles are utilized to estimate abnormal scores for each domain.

4.1 Preprocessing

Here, we describe the procedure to establish a bipartite graph for domain nodes and IP class nodes. First, we transform each hostname into domain names. For example, the domain name of the hostname "www.google.com.tw" is "google.com.tw". Similarly, the domain name of the hostname "www.google.com" is "google.com". Afterwards, we compute the number of times that a domain name h_k adopts an IP class c_i, which corresponds to the element $A(h_k,c_i)$ in k^{th} row and i^{th} column of A. Consequently, by repeating this computation, we complete the matrix A.

4.2 Proxy Filtering

Proxy servers act as an intermediary for clients who connect to proxy servers to request resources available from different servers on the internet.

Traffic logs with proxies involved will mislead the procedure of mutual association discovery. Hence, the removal of proxy related information is required. To achieve this, we observe the behavior reflected from anomaly connections caused by proxy servers in traffic logs. Figure 4.1 shows an

example of associations between domain names and IP classes. Based on the distribution of adopted IP classes, domain names can be divided into two clusters, domain names belonging to Google Inc. and those belonging to Facebook Inc.. On the other hand, based on registered domain names, IP classes can be divided into two clusters, IP classes belonging to Google Inc. and those belonging to Facebook Inc.. In addition, we also find some IP classes (e.g., "216.239.122" in the example), who connect to more than one type of domain names (e.g., both Google and Facebook Inc. domain names) which are confirmed as proxy servers. Such observation inspires us to develop our proxy filtering algorithm as described below.

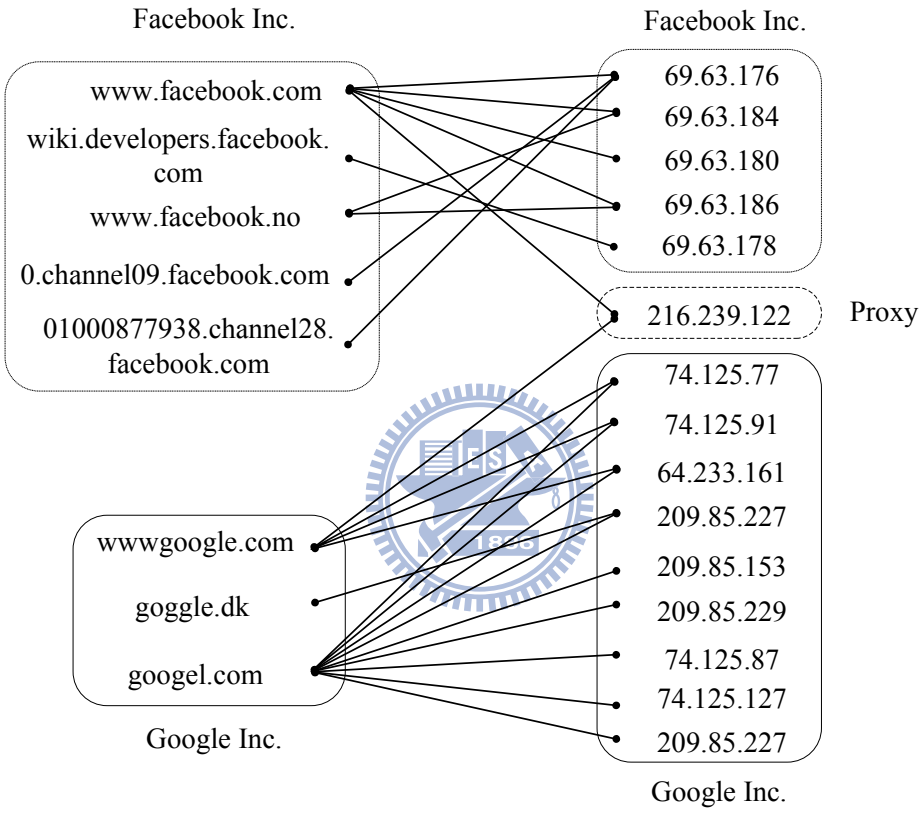


Figure 4.1: An example of proxy detected by mutual associations.

The heart of proxy filtering is to measure the total contributions contributed by each adjacent domain name to an IP class. Algorithm 1 and 2 show the full pseudo code to parallelized measure proxy score for each IP class.

There are two stages in Algorithm 1 and 2. Suppose we want to find the proxy score for a given IP class, c_i. Let $A(:,c_i)$ be the i^{th} column of A. In other words, $A(:,c_i)$ is a vector with each element A_{h_k,c_i} equivalent to number of adoptions between IP class c_i and domain h_k. Similarly, let $A(h_k,:)$ be the k^{th} row of A. Let $sum(A(h_k,:))$ be the summation of each element in $A(h_k,:)$. First, we can compute the contribution of an adjacent domain name h_k to an IPC c_i as follows:

$$P_{i,k} = \frac{sum(A(h_k, :))}{A(h_k, c_i) + smooth \times sum(A(h_k, :))} \tag{4.1}$$

where $smooth$ is a smoothing factor which can control the rate of growth as $A(h_k, c_i)$ decreases. The maximum value is $\frac{1}{smooth}$ when $A(h_k, c_i)$ reaches zero.

Afterwards, for an IP class c_i, the overall proxy score is estimated as:

$$P_i = \prod_{k=1}^{k=|H|} P_{i,k}. \tag{4.2}$$

Basically, an IP class c_i obtains higher proxy score if 1)an IP class node c_i is adjacent to more domain nodes, and 2) the frequency that h_k adopts c_i is lower.

Algorithm 1 $ProxyFilteringMapper(k, V)$

1: $k \equiv$ a domain name h_k
2: $V \equiv$ the k^{th} row of A, $A(h_k, :)$
3: **for** each element $A(h_k, c_i)$ in $A(h_k, :)$ **do**
4: Compute the proxy score $P_{i,k}$ by equation 4.1
5: Emit $\langle c_i, P_{i,k} \rangle$
6: **end for**

Algorithm 2 $ProxyFilteringReducer(k, V)$

1: $k \equiv$ an IP class c_i
2: Initialize $P_i \leftarrow 0$
3: **for** each proxy score $P_{i,k} \in V$ **do**
4: $P_i \leftarrow P_i \times P_{i,k}$ by equation 4.2
5: **end for**
6: Emit $\langle c_i, P_i \rangle$

We evaluate the accuracy of our proxy filtering algorithm by performing Network Mapper (nmap) scanning tests. Nmap is a network scanning, auditing and penetration tool. It determines which hosts are alive on a network and what services and ports are available or open. By using nmap to scan discovered host, we can verify whether the scanned host is a proxy or not. The detail experiments will be reported and discussed in Section 5.

4.3 Mutual Associations Discovery

The key idea in our detection framework is to discover mutual associations based for all pairs of IP classes. A mutual association is relied on two relations:

- domain names belong to an organization usually use a set of similar IP classes, and
- a set of IP classes usually used by domain names of the same organization.

Table 4.1 illustrates an example of IP classes that two domain names adopted as well as their adoption counts. In this case, the two hostnames "www.google.com" and "www.gmail.com" that created by Google Inc. but with different domain names still shares high similarity in their adoption list of IP class. Similarly, some IP classes can only used by specific organizations. Figure 4.2 demonstrates an example of domain clusters and IP class clusters. A set of domains that share similar traits at their domain names is hardly to distinguish which belongs to Google Inc. and which is not. However, by exploiting the associations between domains and IP classes, the set of domains can be divided into "Google Inc." cluster and "Non Google Inc." cluster based on their adoption history on IP classes. Instead of clustering from the perspective of textual analysis, this example illustrates the importance of mutual associations in distinguishing abnormal domains.

Table 4.1: Top five IP classes adopted by "www.google.com" and "www.gmail.com".

Hostname	74	72	66	64	209.85.165	209.85.133	209.85.137	216.239.61
www.google.com	**1095**	**1095**	**1095**	**1095**	**1094**	1089	0	1083
www.gmail.com	**1075**	**900**	102	265	38	**503**	**330**	**306**

The key point is to construct a proximity matrix for IP class nodes, which in turn can be cast as finding a proximity function to measure associations between any two IP classes.

Algorithm 3 and 4 list detained procedures to compute the proximity matrix. Overall, we adopt the idea of mutual information for this purpose. The degree of mutually associated between two IP class c_i and c_j with related to a domain name h_k is formalized as follows:

$$M_{i,j,k} = A(h_k, c_i) \times A(h_k, c_j). \tag{4.3}$$

In general, the value of $M_{i,j,k}$ is higher if 1) both c_i and c_j are adopted by h_k, and 2) the frequencies of respective adoptions by h_k are higher. Besides, if at least one IP classes is not adopted by h_k, $M_{i,j,k}$ goes zero.

Afterwards, for a pair of IP classes c_i and c_j, their mutual association score is computed by aggregating $M_{i,j,k}$ over domain space H as follows:

$$M_{i,j} = \sum_{h_k \in H} M_{i,j,k}. \tag{4.4}$$

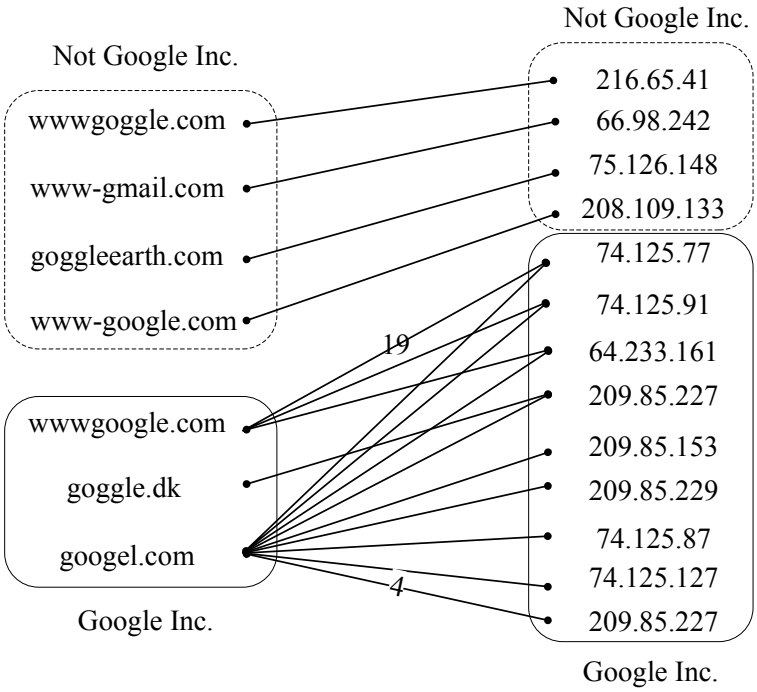


Figure 4.2: An example of co-clusters identified by mutual associations between domain names and IP classes.

Mutual association ranges from 0 to ∞. Not that mutual association score is symmetric. Basically, a pair of IP classes c_i and c_j has high mutual association if they share similar adoption histories. In other words, if they are frequently adopted by similar lists of domains, they are mutually more associated.

4.4 Abnormal Domain Estimation

The goal of abnormal detection for a given domain name is to estimate the degree of being anomaly with its mapping address. To achieve this, we design an anomaly function to map a given domain name to a score by using discovered mutual associations. The basic idea of anomaly score computation is to measure the diversity of mutual associations between any pair of IP classes that a domain is adjacent to. In other words, if the set of IP classes that a domain adopted diverges from IP classes such that each pair of which are less associated to each other, then the domain has high possibility to be an abnormal domain. Another extreme case is when the set of IP classes that a domain adopted converges on a set of IP classes each of which are highly associated, the domain is considered benign.

The anomaly score computation can be cast as bridging score computation in graph mining. Bridging score measures the degree of connectivity to other groups on a graph. Betweenness [12][13][14] is designed to measure the degree of connectivity. However, as discussed in [15],

algorithms for computing betweenness require a considerable computation costs. The authors in [16] consider the degree of connectivity of a node on a graph as a metric for evaluating anomaly score of a node on a graph. Instead, they make use of relevance score of neighborhood of a target node for efficient anomaly score computation. We borrow that idea for anomaly score computation and design our anomaly score based on mutual association score discussed above as follows.

Given a collection of suspicious DNS records, let S denotes the matrix where each entry $S(h_k, c_i)$ indicates how many time the suspicious domain h_k adopts the IP class c_i. The parallel mutual association score selection for each DNS records is described below and provide the pseudo code in Algorithm 5 and Algorithm 6. Particularly, regarding a domain name h_k, let its adopted IP classes as $N_{h_k} = \{c_i | S(h_k, c_i) \geq 0\}$. Then, we select the mutual association scores between any pair of elements $M(i, j)$ for c_i and c_j in N_h.

After that, we take the summation of all pairs of mutual association score as follows:

$$\widehat{R}(N_{h_k}) = \sum_{c_i, c_j \in N_{h_k}, c_i \neq c_j} M_{i,j}. \tag{4.5}$$

Finally, we obtain the bridging score by dividing one by average $\widehat{R}(N_{h_k})$ as follows:

$$bs(h_k) = \frac{\binom{|N_{h_k}|}{2}}{1 + \widehat{R}(N_{h_k})}. \tag{4.6}$$

Note that the value of bridging score is between 0 and $\binom{|N_{h_k}|}{2}$. A domain has high bridging score if 1) the domain adopts many IP classes (i.e., N_{h_k} is large), and 2) summation of all pairs of mutual association is small.

Algorithm 3 $MutualAssociationMapper(k, V)$

1: $k \equiv$ a domain name h_k
2: $V \equiv$ the k^{th} row of A, $A(h_k,:)$
3: **for** each element $A(h_k, c_i)$ in $A(h_k, :)$ **do**
4: **for** each element $A(h_k, c_j)$ in $A(h_k, :)$ and $c_i \neq c_j$ **do**
5: Compute the mutual association score $M_{i,j,k}$ by equation 4.3
6: Emit $\langle c_i, (c_j, M_{i,j,k}) \rangle$
7: **end for**
8: **end for**

Algorithm 4 $MutualAssociationReducer(k, V)$

$k \equiv$ an IP class c_i
Initialized $M_{i,j} \leftarrow 0$ for all c_j in values of V
for each map value $(c_j, M_{i,j,k}) \in V$ **do**
    $M_{i,j} \leftarrow M_{i,j} + M_{i,j,k}$ by equation 4.4
end for
for each IP class c_j associated to c_i **do**
    Emit $\langle c_i, (c_j, M_{i,j})\rangle$
end for

Algorithm 5 $BridgingScoreMapper(k, V)$

Let $k_H \equiv$ a suspicious domain name h_k
Let $V_H \equiv$ the k^{th} row of suspicious DNS record matrix, $S(h_k,:)$
Let $k_C \equiv$ an IP class c_i
Let $V_C \equiv$ the i^{th} row of mutual association matrix, $M(c_i,:)$
for each element $S(h_k, c_i)$ in V_H **do**
    Emit $\langle c_i, h_k)\rangle$
end for
for each element $M(c_i, c_j)$ in V_C **do**
    Emit $\langle c_i, c_j)\rangle$
end for

Algorithm 6 $BridgingScoreReducer(k, V)$

$k \equiv$ an IP class c_i
Let $V_H \equiv$ the set (S(:,c_i), $M(c_i,:)$)
Let $V_C \equiv$ the i^{th} row of mutual association matrix, $M(c_i,:)$
for each value $(h_k, c_j) \in V_H$ **do**
    Emit $\langle h_k, M(i, j)\rangle$
end for

Chapter 5

Experimental Results

In this section, we give experimental results for our abnormal domain detection framework. Our evaluation mainly focuses on 1) the quality study and 2) the scalability study. We gives a brief introduction of the data set. Then, we report the experimental results of proposed proxy filtering algorithm, mutual association discovery and abnormal domain detection. Finally, we demonstrate the scalability of our framework with various input size and number of computing nodes in distributed environment.

5.1 Data set

We conduct our experiments by using network traffic logs from Trend Inc. and NCTU dormitory respectively. Overall, there are 256M events in Trend data set in one day. NCTU dormitory traffic logs contain 90M events are collected during Nov. 6-17, 2009. The detail statistics about the data sets are summarized in Table 5.1

Table 5.1: Dataset Statistics

	NCTU_HTTP	Trend_HTTP
Number of Logs	90,830,834	256,815,644
Distinct destination IP	158,432	51,6176
Distinct destination IP class	43,694	142,141
Distinct destination hostname	176,529	1,733,899
Distinct source IP	9,963	295,197
Duration	1 week (11/6 - 11/17)	1 day (5/19)

5.2 Quality Study

In this section, we evaluate the performance of our proxy filtering algorithm and abnormal domain detection in terms of accuracy.

5.2.1 Accuracy of Proxy Filtering

Evaluation Metrics

To evaluate the performance of our proxy filtering algorithm, we compute a precision metric as follows.

- Precision : Number of correctly detected IP address among all IP address returned by our approach

In our experiment, we return a list of IP address as well as its port detected by our approach. Then, we verify each of them by performing nmap (Network Mapper) scanning test. Namp takes an IP address and a port to discover services, software product used to run a service, exact version number of that product etc. on a scanned host. By manual analysis from the software product used to run a service responded from a scanning test, we can confirm whether a scanned IP address is a proxy server or not.

Table 5.2.1 shows an example of returned information by nmap scanning test during our experiment. Some software products such as ”AkamaiGHost”, ”nginx http proxy”, ”Squid webproxy”, and ”Sandpiper Footprint http load balancer” are labeled proxy-relevant. If a target IP class is running one of the proxy-relevant software products in the list, then the IP class is confirmed as a proxy server.

Results

Figure 5.1 and Figure 5.2 show the precision as function of the length (up to 100) of the returned list for proxy filtering with and without smoothing factor included on different data sets. We set $smooth$ to 0.1. As precision of filtering with smoothing factor is nearly twice higher than that of filtering without smoothing factor, we can conclude that proxy filtering with the introduce of smoothing factor achieves more effective detection accuracy than that of non-smoothing case. This is because by taking smoothing factor into consideration, proxy scores will not be dominated by low-frequency adoption of IP classes.

In most cases, the precision is around 70% achieved by smoothed proxy filtering on NCTU dormitory data set in Figure 5.1. During evaluation, some scanning tests are failed which might owing

Table 5.2: A list of software products collected by performing nmap scanning test.

Software Product	Proxy Or Not
3Com switch webadmin	No
AkamaiGHost	Yes
AOLserver httpd	No
Apache httpd	No
Apache Tomcat/Coyote JSP engine	No
Caucho Resin JSP engine	No
Google httpd	No
hp color LaserJet 4650	No
HP Jetdirect httpd	No
HP LaserJet	No
IBM HTTP Server	No
Icecast http statistics plugin	No
Jetty httpd	No
lighttpd	No
Microsoft IIS webserver	No
Microsoft Windows Media Server	No
Netscape Enterprise httpd	No
Netscreen administrative web server	No
nginx http proxy	Yes
Sandpiper Footprint http load balancer	Yes
Squid webproxy	Yes
SunONE WebServer	No
Urchin RSS aggregator	No
Zeus httpd	No

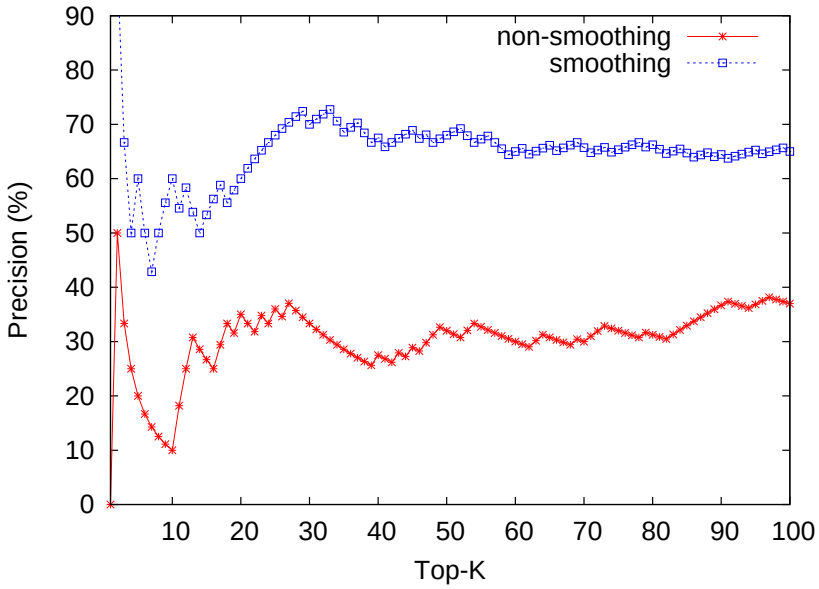


Figure 5.1: Precision of proxy filtering on NCTU dormitory data set.

to the downtime of scanned host. In that case, even though an IP class behaves like a proxy server in collected raw traffic logs, it will be considered a false alarm in our experiments. Consequently, it causes the decrease of detection accuracy. Overall, in the collected proxy pool, 298 (around 24.9%) IP classes lack of software product information.

5.2.2 Accuracy of Mutual Association Discovery

To evaluate the performance of proposed mutual association discovery algorithm, we collect IP classes for well-known corporations on the internet. Afterwards, to demonstrate the effectiveness of mutual associations for distinguishing domains, examination of mutual associations for IP classes within a company (e.g., Google) and between two different companies is performed. Specifically, in the case of intra-company experiment, we collect the IP classes adopted by each company. Then, for each pair of IP classes corresponding to a company, we compute its mutual associations. Finally, we collect all pairs of computed mutual association scores and draw a distribution for that. In the case of inter-company experiment, for each pair of companies, we compute mutual associations by iteratively taking one IP class from respective set of IP classes. Finally, we collect all pairs of computed mutual association scores and draw a distribution for that.

A public AS has a globally unique number, an ASN, associated with it. This number is used both in the exchange of exterior routing information (between neighboring ASes) and as an identifier of the AS itself. Following the same way, we take ASN as the groundtruth to evaluate mutual association algorithm. Detail statistics of groundtruth for evaluation are summarized in Table 5.2.2. As illustrated in Table 5.2.2, only a few IP classes under a company appear in our dataset. Hence,

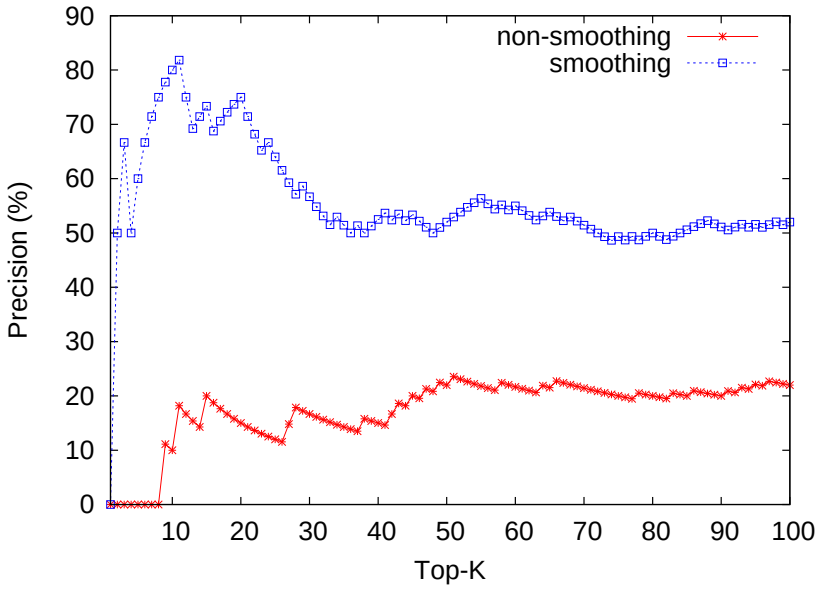


Figure 5.2: Precision of proxy filtering on Trend Inc. data set.

a group of IP classes under an ASN may not share similar characteristics. For example, some serve as PC, others may serve as web servers. Although the associations between IP classes and company can be aware via ASN, we still cannot learn that which pair of IP classes are mutually associated and which pairs are not.

Table 5.3: Groundtruth Statistics.

Company	#AS	#Prefix	#IP Class	#IP Class Appeared in our Dataset
Google	5	27	875	127
Facebook	1	4	40	10
Yahoo	34	177	5874	406

Evaluation Metrics

We examine the differences of mutual association scores by illustrating distributions of mutual associations for IP classes inter(intra) company and inter(intra)ASNs.

- IntraComp : Distribution of mutual association scores for IP classes from the same company
- InterComp : Distribution of mutual association scores for IP classes from different companies
- IntraASN : Distribution of mutual association scores for IP classes from the same ASN
- InterASN : Distribution of mutual association scores for IP classes from different ASNs

Results

Figure 5.3 and Figure 5.4 illustrate comparison for mutual associations for IP classes and ASNs respectively. As we expected, the mutual associations among a group of IP classes under an ASN are higher than that under a company. The results prove the effectiveness of proposed mutual associations.

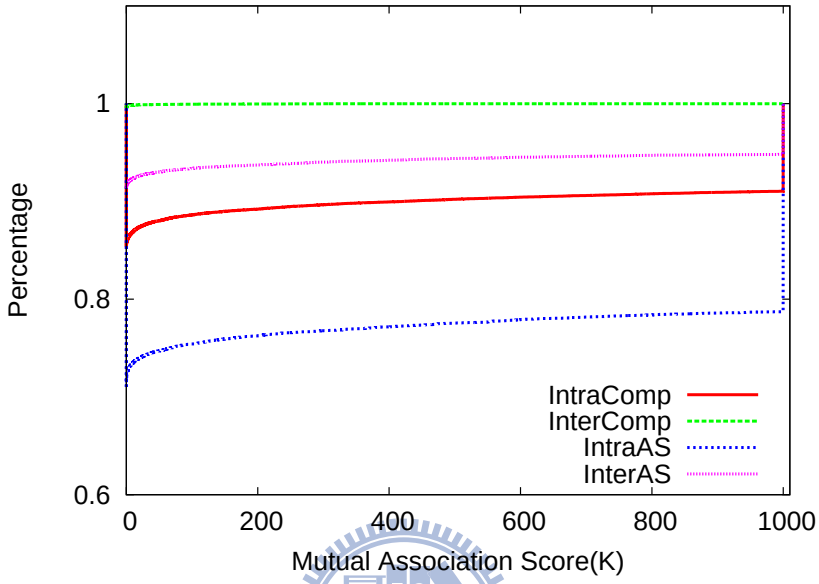


Figure 5.3: Evaluation on mutual association scores (Trend).

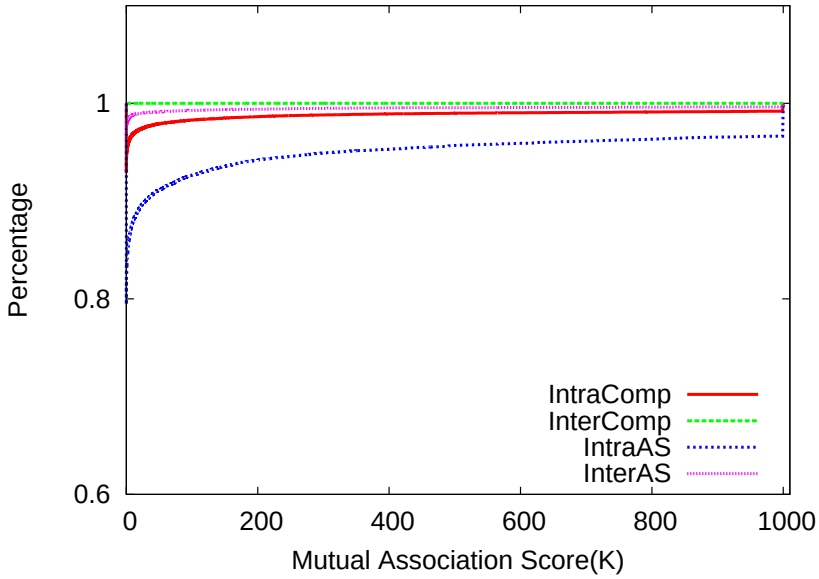


Figure 5.4: Evaluation on mutual association scores (NCTU).

5.2.3 Accuracy of Abnormal Domain Detection

To evaluate the performance of proposed framework for abnormal domain detection, we conducted experiments for FFSN detection. We use the domain corpus from the website [17]. We use both benign and suspected domains. Overall, "lookups-benign" corpus contains 34,647 benign domains; "lookups-ndss-ff" corpus contains 94 suspected fast-flux domains detected by [4]. "lookups-ndss-ff" corpus serves as a groudtruth for evaluating the performance of proposed model. Specifically, we compute bridging scores for all unique domains in collected corpus based on constructed profiles, which results in a list of domains ranked by bridging scores. By computing precision and recall for different length of returned domains, we complete the performance evaluation on FFSN detection.

Evaluation Metrics

We compute the following metric to evaluate the effectiveness of FFSN detection.

- Domain-Precision : Number of correctly detected domains among all domains detected by our approach.
- Domain-Recall : Number of correctly detected domains among all domains in "lookups-ndss-ff" corpus.

Results

Figure 5.5 shows the quality of ranking using the recall-precision graph for FFSN detection. We construct two set of profiles based on Trend Inc. and NCTU dormitory data set respectively. In top 89 domains returned by our model, all returned domains are confirmed as FFSN domains as [4] did. The top 95 returned domains contain one benign domain (bs("runescape.com")=66 at 90^{th}). Table 5.4 shows the detail information of "runescape.com". The reason why "runescape.com" is regarded as a suspect is because the mutual association for each pair of its IP classes learnt from the similarity of their neighboring domains in both data set is zero, which results in a large bridging score. Note that zero mutual association indicates the adjacent domains of two IP classes are disjoint, which means the two IP classes never co-occur in any traffic log.

Table 5.4: The IP classes adopted by "runescape.com."

64.37.71	64.90.181	65.39.250	66.151.43	69.22.158	69.31.109
80.64.4	82.133.85	85.133.44	168.75.179	209.249.24	216.180.254

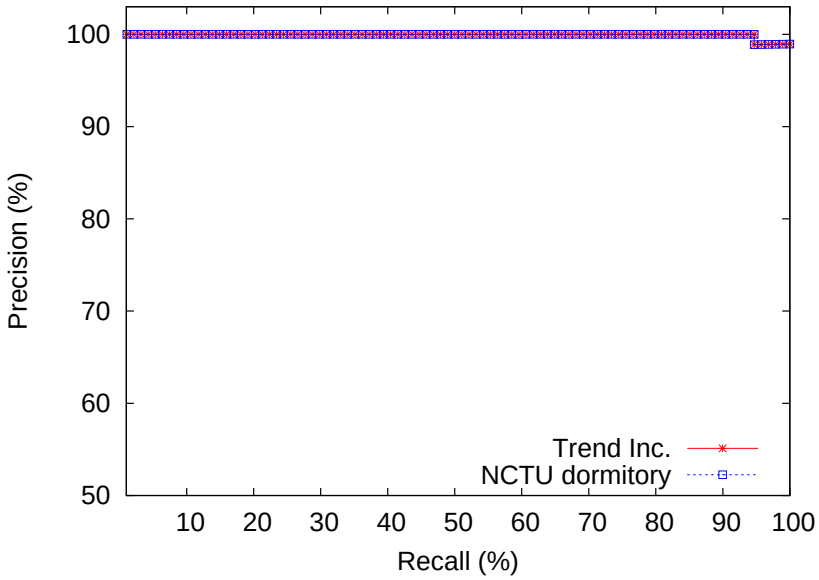


Figure 5.5: Recall-precision graph of the abnormal domain detection model for FFSN corpus.

5.3 Scalability Study

5.3.1 Experimental Settings

For the computational cost, we report the wall-clock time. The experiments ran on five machines with each 2.66GHz dual-core CPUs and 4GB memory, running Debian Linux. Particularly, we measure the running time for abnormal domain detection with 1) different size of computing nodes, and 2) different amount of traffic logs.

In analyzing runtime speedup for parallel profile construction, we learn a abnormal domain detector on two machine to six machines. We use the runtime of one machine as the baseline and assume that six machine can achieve six time speedup.

5.3.2 Results

Figure 5.6 reports the runtime speedup of using up to six machines. It takes less time to build model with more computing nodes. The main reason why more computing nodes ensure more efficiency is because the architecture of our distributed file system. We use six machines to construct both hadoop distributed file system and map/reduce paradigm. Therefore, when there is more computing nodes available, more I/O operation can be performed locally instead of remotely.

Figure 5.7 reports the runtime comparison for different amount of input size. When the data size increases, the runtime growth linearly. On the other hand, when the number of computing nodes increases, the runtime required decreases linearly. As a result, our approach can be applied to larger dataset if we can construct an environment with more computing nodes.

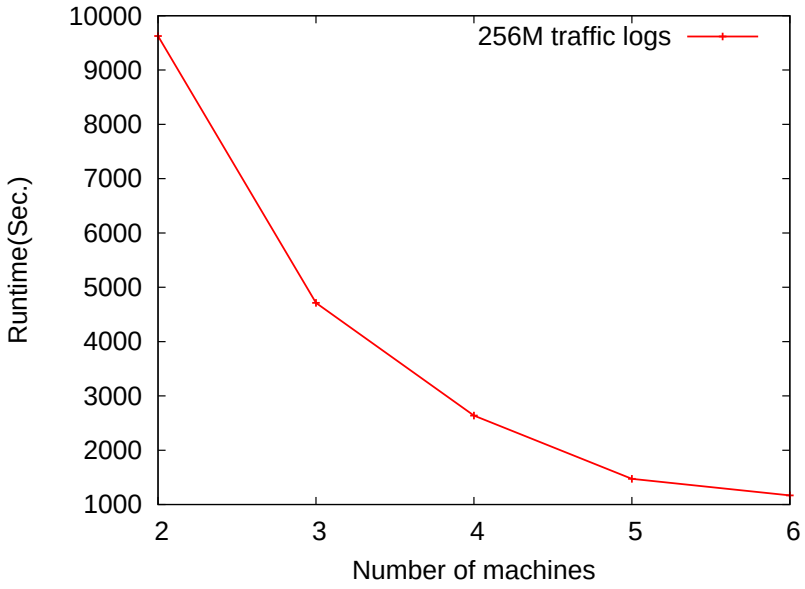


Figure 5.6: Runtime comparison for different number of machines.

Figure 5.8 reports runtime composition for several jobs in building detection model. The running time consists of three man parts: 1) construction mappings from domains to IP classes, 2) construction of mutual association matrix between IP classes, and 3) proxy filtering. The major computations in building detection model are construction mappings from domains to IP classes and proxy filtering. In addition, it demonstrates the linear growth in runtime from two sizes of traffic logs.

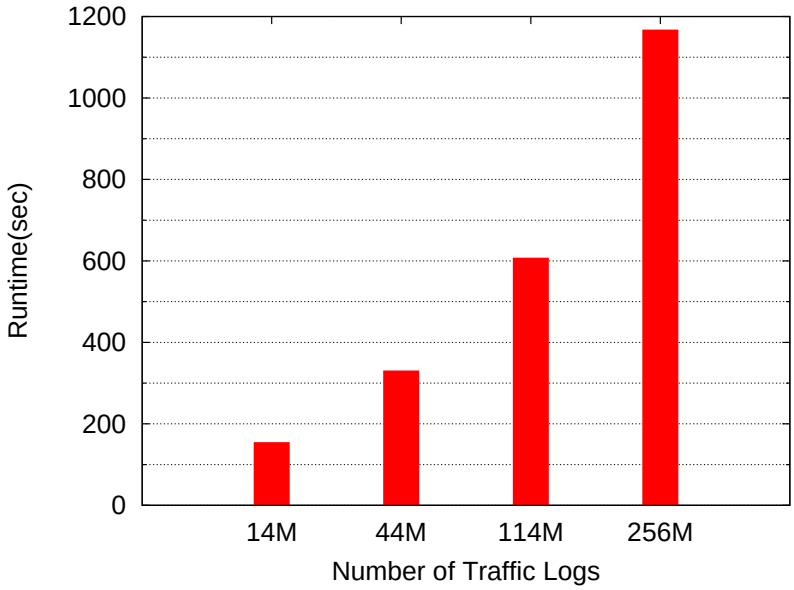


Figure 5.7: Runtime comparison for different sizes of traffic logs.

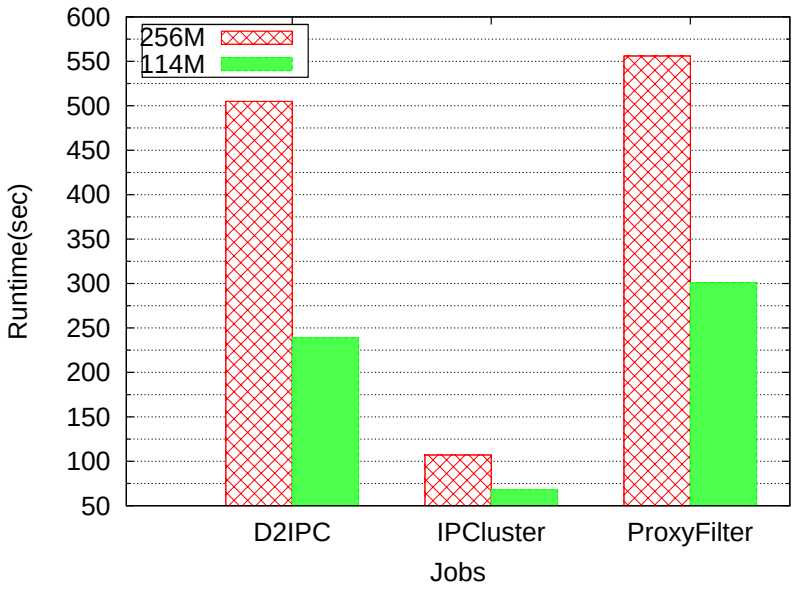


Figure 5.8: Runtime composition analysis on 114M and 256M traffic logs.

Chapter 6

Conclusion and Future Work

Due to changes in the propagation and update strategy, abnormal domain detection by modeling rapid IP changes is insufficient. Classical fast-flux technique is used to control a botnet and establish communication channels by changing IP addresses. However, Conficker goes to another level by including a pseudo-random domain name generator to last the communication channel longer between a control node and its botnet.

In this study, we propose an automatic framework for abnormal domain detection. To identify abnormal domains generated by fast-flux technique or domain name generators, we propose to discover mutual associations between domain names and IP address. In particular, profile construction contains two procedures to discover mutual associations between domain and IP address. Firstly, a proxy filtering algorithm is proposed to remove proxy servers from collected IP addresses. Secondly, a co-clustering algorithm is proposed to partition domain and IP class node into clusters. Afterwards, the procedure for abnormal domain name detection which computes an anomaly score for a given domain according to discovered co-clustering information is performed.

The main contributions of the study includes:

- The development of approach for discovering mutual associations.
- The design of parameter free framework for abnormal domain detection by exploiting mutual associations.
- The development of approach for proxy detection algorithm by exploiting co-occurrences among domains

This work suggests several interesting directions for future work. In addition to the problem of predicting an anomaly score for a given domain, a prediction system for abnormal domain names is also an useful alarm tool when the real traffic streams involved with predicted domain names

haven't yet been existed. The anomaly concept we considered in this study is temporal-insensitive. However, some abnormal domains can vary to another one once they are blocked. To trace variation patterns of abnormal domains over time for abnormal domain prediction is an interesting direction for future work. Besides, the approaches for proxy filtering and abnormal domain detection proposed in the study show the importance of exploiting associations between IP class and domain names. In other words, the clustering results of IP class can effect the detection of abnormal domains; the clustering results of domain names can effect the detection of proxy IP class. However, currently the associations are adopted separately. To design an iterative and interactive co-clustering algorithm for proxy detection and abnormal domain detection simultaneously is also an interesting and challenging direction for future work.

Bibliography

[1] "Fast flux botnet data service," http://wiki.milcordme.com/wiki/Fast_Flux_Botnet_Data_Service, 2009.

[2] "Ssac advisory on fast flux hosting and dns," in *ICANN Security and Stability Advisory Committee (SSAC)*, 2008.

[3] Wikipedia, "Conficker," http://en.wikipedia.org/wiki/Conficker#Automated_remote_detection, 2009.

[4] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks," in *NDSS '08: Proceedings of The 16th Network and Distributed System Security Symposium*, 2008.

[5] J. Park, "W32.downadup.c pseudo-random domain name generation," http://www.symantec.com/connect/blogs/w32downadupc-pseudo-random-domain-name-generation#A258, 2009.

[6] D. Dittrich and S. Dietrich, "Discovery techniques for p2p botnets," in *Stevens CS Technical Report*, 2008.

[7] E. Stinson and J. C. Mitchell, "Towards systematic evaluation of the evadability of bot/botnet detection methods," in *WOOT '08: Proceedings of The Second Conference on USENIX Workshop on Offensive Technologies*, 2008.

[8] J. B. Grizzard, V. Sharma, C. Nunnery, B. B. Kang, and D. Dagon, "Peer-to-peer botnets: overview and case study," in *HotBots '07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*. USENIX Association, 2007.

[9] P. Barford and V. Yegneswaran, "An inside look at botnets," *Advances in Information Security*, vol. 27, pp. 171–191, 2007.

[10] R. McMillan, "Experts bicker over conficker numbers," http://news.techworld.com/security/114307/experts-bicker-over-conficker-numbers/, 2009.

[11] A. Kriegisch, "Detecting conficker in your network," http://www.cert.at/static/conficker/TR_Conficker_Detection.pdf, 2009.

[12] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[13] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, pp. 163–177, 2001.

[14] T. H. Haveliwala, "Topic-sensitive pagerank," in *WWW '02: Proceedings of the 11th international conference on World Wide Web*, 2002.

[15] K. Onuma, H. Tong, and C. Faloutsos, "Tangent: a novel, 'surprise me', recommendation algorithm," in *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[16] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Relevance search and anomaly detection in bipartite graphs," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 48–55, 2005.

[17] "Pi1 - fast flux," http://pi1.informatik.uni-mannheim.de/fast-flux, 2008.