Missouri S&T
Library and Learning Resources

Scholars' Mine

Masters Theses

Student Theses and Dissertations

Fall 2008

Critical infrastructure protection and the Domain Name Service (DNS) system

Mark Edward Snyder

Follow this and additional works at: https://scholarsmine.mst.edu/masters_theses

Part of the Computer Sciences Commons

Department:

Recommended Citation

Snyder, Mark Edward, "Critical infrastructure protection and the Domain Name Service (DNS) system" (2008). *Masters Theses*. 4638.
https://scholarsmine.mst.edu/masters_theses/4638

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.

المنارة للاستشارات

www.manaraa.com

CRITICAL INFRASTRUCTURE PROTECTION AND THE DOMAIN NAME SERVICE (DNS) SYSTEM

by

MARK E. SNYDER

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN COMPUTER SCIENCE

2008

Approved by

Dr. Bruce McMillin, Co-advisor
Dr. Mayur Thakur, Co-advisor
Dr. Ann Miller

المنارة للاستشارات

www.manaraa.com

Copyright 2008
Mark E. Snyder
All Rights Reserved

www.manaraa.com

PUBLICATION THESIS OPTION

This thesis consists of the following two articles that have been submitted for publication as follows:

Pages 1–25 have been accepted for publication as: Mark E. Snyder, Ravi Sundaram, Mayur Thakur, “A Game-Theoretic Framework for Bandwidth Attacks and Statistical Defenses,” *lcn*, pp. 556-566, 32nd IEEE Conference on Local Computer Networks (LCN 2007), 2007.

Pages 26–52 are intended for submission to the IEEE ICC 2009 COMMUNICATION AND INFORMATION SYSTEMS SECURITY (CISS) SYMPOSIUM.

المنارة للاستشارات
www.manaraa.com

ABSTRACT

Components of the critical infrastructure of any system are natural targets for attack. Any inherent weakness of such components can potentially expose the entire system to vulnerability. The Domain Name System (DNS) is one component of the proper functioning of the Internet. Although DNS is a relatively simple, isolated component, it serves as a straightforward example for the study of distributed systems in general, and as such, we have explored properties of DNS to examine how enterprise-scale, critical infrastructure components are vulnerable to attack, what protections are afforded to defenders of such components, the inherent weaknesses of such systems, and what natural defenses are available to safeguard them and ensure the availability of these systems. Each of the works in this thesis analyzes some aspect of our efforts in this regard.

In "Preprocessing DNS Log Data for Effective Data Mining," we focus on the task of obtaining DNS log data and the task of data preparation and cleaning to place the data in a form which can be data mined. In this effort, the problem of data insufficiency required a non-standard approach to data cleaning. We infer missing values by exploiting business knowledge of DNS behavior, preserving features lost using methods such as linear interpolation.

In "A Game-Theoretic Approach to Bandwidth Attacks and Statistical Defenses," we have discovered a novel way of framing a bandwidth attack as a competitive, two-player game. There exist more definitive methods than ours for sifting through DNS traffic that can isolate and take measures to deal with illegitimate requests, but these methods consume more resources in doing so than most DNS servers choose to spare. As a result, the typical response to a spike in illegitimate DNS requests is simply to add more server capacity. Our method shows that it is possible to identify groups of traffic that are suspicious. It is hoped that by using a technique such as ours as a front-line filter to throttle the requests sent on for more definitive analysis, we might make the utilization of the more definitive techniques more attractive, thereby protecting the DNS servers from becoming overloaded with requests, thereby protecting the proper functioning of this service.

المنارة للاستشارات

www.manaraa.com

ACKNOWLEDGMENT

I would like to take a moment to thank Dr. Mayur Thakur for exposing me to his infectious passion for computer science, both in the classroom and in research. Thanks also to Dr. Bruce McMillin for his guidance, assistance and encouragement, as well as his patient instruction in all things procedural. Additional gratitude is due to Dr. Ann Miller for her participation and useful discussions on research topics throughout. Thank you to Dr. Ravi Sundaram for his hospitality, support, and for our discussions on my research.

I would also like to thank my parents, Mackey and Carol Snyder, for instilling in me the desire to continually pursue knowledge, for being examples of how to focus my efforts, and for encouraging me to develop the tenacity to see things through. My late father-in-law James D. Kolb was also ever-present in my thoughts during this effort. He would have relished this achievement as much as do I.

Finally, I would like to thank my wife, Sandi. No project like this can be completed without sacrifice on many fronts in one's life. Sandi worked as hard as I did to make this possible. Her efforts, however, were expended not in front of the keyboard, but in patiently listening to me discuss my work and by picking up the slack at home.

المنارة للاستشارات
www.manaraa.com

TABLE OF CONTENTS

المنارة للاستشارات

www.manaraa.com

المنارة للاستشارات

www.manaraa.com

LIST OF ILLUSTRATIONS

المنارة للاستشارات
www.manaraa.com

المنارة للاستشارات

www.manaraa.com

LIST OF TABLES

www.manaraa.com

PAPER

I. A GAME-THEORETIC FRAMEWORK FOR BANDWIDTH ATTACKS AND STATISTICAL DEFENSES

We introduce a game-theoretic framework for reasoning about bandwidth attacks, a common form of distributed denial of service (DDoS) attacks. In particular, our *traffic injection game* models the attacker as a rational but limited-resource entity who uses limited knowledge of traffic patterns to launch IP spoofing based bandwidth attacks on a server. We model the defender as a coarse-grained, relative volume based statistical filter.

We analyze the effectiveness of the defender against the attacker by analyzing the payoffs of various strategies in the traffic injection game. Furthermore, we analyze how these payoffs change in the presence of random noise.

Our results show that there is potential for using statistical methods for creating defense mechanisms that can detect a DDoS attack and that even when an attacker has a priori knowledge of the expected traffic volume for the dimension and divisions employed in the attack, the attack traffic can still be exposed to the defender.

المنارة للاستشارات
www.manaraa.com

1. INTRODUCTION

IP spoofing—changing the source address of IP packets—has been used in DDoS attacks on popular websites (Yahoo!) and root DNS servers. Examples of DDoS attacks that use IP spoofing are Smurf and SYN attacks. IP spoofing-based attacks are dangerous because they can bypass filters that drop packets based on IP addresses. Even though techniques such as backscatter can detect IP spoofing-based attacks, one needs more sophisticated filtering mechanisms to detect more intelligent attacks. However, these filtering mechanisms cannot be too fine-grained because they have to be installed on routers. Also, simple volume-based detection mechanisms are unlikely to be effective because of the busyness of the Internet traffic. We thus study the effectiveness of *relative volume* and *coarse-grained* measures against IP spoofing attacks.

During a Distributed Denial of Service (DDoS) attack, the attacker increases the amount of illegitimate traffic originating from machines under his control. This results in a positive increase by some ratio relative to the traffic that was present in the system to begin with. We will refer to this ratio as α, where $0 \leq \alpha \leq 1$.

We view the attacker as choosing a distribution by which traffic is added to the system by the machines under his control. The attacker can view this from a number of directions, which we will call dimensions. For example, an attacker might choose to have all controlled machines use only spoofed IP addresses that are even-numbered. In this case, divisibility by 2 becomes a dimension, and there are 2 divisions within this dimension, even and odd. Traffic can be increased using a geographical distribution. The attacker may decide to increase traffic only to target machines in New York, or to add twice as much traffic coming from New York machines as from Los Angeles machines. Thus the geographical location of machines becomes a dimension and there could be numerous divisions within this dimension. The attacker is able to choose a distribution across the divisions of any dimension he envisions. In any set the number of groupings is only limited by the power set, which, while very large is still finite. In

المنارة للاستشارات
www.manaraa.com

each case, the attacker identifies a dimension and then chooses a distribution across the divisions within that dimension.

The defender views the traffic in much the same way. As traffic is received, the defender counts it and monitors how the traffic is distributed amongst the divisions of some dimension. If the defender can detect a change from the base distribution, then it knows that an attack may be underway. The challenge for the defender, however, is that even though it can detect that an attack may be happening, there is still the matter of sorting the legitimate requests from the illegitimate. The solution that we will propose is to merely tag each division within the observed dimension with a measurement of suspiciousness. The more suspicious a group of traffic is, the more scrutiny it will be subjected to and thus the longer it will take to process, but it will not be dropped. Whereas traffic that is not suspicious is not delayed at all. Therefore, the defender in our game must only identify the groups of traffic that are more suspicious than other groups of traffic.

This is a very different approach than, say, that employed by synkill[8] where the goal is to spend as few resources as possible on suspicious traffic, but our approach is meant as a first line filter to incoming traffic, and the goal is to advise subsequent processes about which traffic merits additional scrutiny using more definitive techniques.

Since neither the attacker nor defender has perfect information about all dimensions and divisions within those dimensions, we define a “hidden” distribution of traffic to represent these hidden dimensions. We take care to note that our model is equivalent to one that allows the attacker to control, defender to monitor, multiple dimensions as these multiple dimensions can be projected onto a new dimension so that it can be represented as a single dimension again. When an attacker adds traffic according to its chosen distribution, the hidden distribution is applied to the traffic across all other dimensions to produce the counts that the defender views from the perspective of his chosen dimension. Thus, even if the attacker is aware of or tries to approximate the base distribution with the attack distribution, the hidden distribution disseminates the traffic in such a way that the defender can still observe an anomaly with a careful choice of dimension to observe.

The Traffic Injection Game is a two person, zero sum, competitive game with

المنارة للاستشارات
www.manaraa.com

imperfect knowledge. We represent the game in normal form as a sensitivity matrix in the form of a bimatrix of summed difference of new traffic to base traffic once the attack traffic is added [9].

Our game consists of two d-dimensional matrices—base distribution (representing the relative volume of traffic distribution) and a hidden distribution (representing the relative entity distribution). The attacker and defender (the two players in the game) each choose (independently) a dimension. We assume that the attacker and the defender both have knowledge of the base distribution along the dimension of their choice. The attacker chooses to inject traffic in the network (based on its knowledge of the distribution along a dimension), while the defender uses statistical means to measure the change in distribution from the normal distribution. The change in distribution per unit of attack traffic injected is the payoff to the defender.

We show that knowledge of traffic distribution helps both the attacker and the defender. When an attacker chooses a strategy (the amount of attack traffic that it generates per division of the attack dimension) that closely matches the base distribution for the attack dimension, then the payoff is best for the attacker. As the strategy deviates more and more from the base distribution, the attack traffic becomes more exposed to the defender.

We also show that even when the attacker knows the base distribution for the attack dimension and utilizes it to its best advantage, the attack is still detectable, given favorable signal-to-noise ratio and a careful choice of dimension by the defender in which to monitor traffic.

المنارة للاستشارات

www.manaraa.com

2. RELATED WORK

A variety of methods of dealing with the problem of denial of service (DoS) attacks have been explored. The idea of comparing incoming to outgoing flows of network traffic was explored in [7]. Using game theory to identify another kind of DoS attack, that attempted by nodes in wireless sensor networks, was investigated in [1]. Cominetti, et al., examined the cost of anarchy in network games[2], following work by Roughgarden and Tardos[6]. Many of the defense mechanisms that have been proposed to combat these attacks are described, and advantages and disadvantages of each proposed scheme are outlined in [3]. In [4] techniques to thwart IP address spoofing in DNS DDoS attacks are detailed in the form of a firewall process, but this work also highlights the disadvantage to such definitive techniques, in that they have the potential to create a bottleneck themselves. The danger of indiscriminately dropping suspected traffic is described in [5]. They compare DDoS attacks to *flash crowds* which look similar to a DDoS attack but constitute legitimate traffic. Schuba, et al., contributed a detailed analysis of the SYN flooding attack and a discussion of existing and proposed countermeasures[8]. Calculating Nash equibria in bimatrix games in normal form was detailed by von Stengel[9].

المنارة للاستشارات
www.manaraa.com

3. TRAFFIC INJECTION GAME

We now define a game which would allow us to analyze IP-spoofing based bandwidth attacks. Our game is a 2-player game. We call the two players *attacker* and *defender*. We will refer to the server under consideration as *the server*. The attacker's motive is to attack the server using an IP-spoofing based bandwidth attack, while the defender's motive is to detect these attacks using coarse-grained statistical filtering.

3.1. BASE AND HIDDEN DISTRIBUTIONS

The coarse granularity of filtering is modeled using a set of *dimensions*. Each dimension is a partition of the IP address space, and a corresponding partition of the total *typical* IP traffic reaching the server. We call the former the *hidden distribution* and the latter the *base distribution*. (The reason for these names will become clear shortly.) For example, if our dimension is the "autonomous system" (AS) dimension, then the ith component of the hidden distribution is the fraction of IP addresses in the ith AS, and the ith component of the base distribution is the fraction of traffic reaching the server that originates in the ith AS.

First, we need to model typical (or *base*) traffic, which we do by defining an n-dimensional matrix of traffic values T. Let d_i denote the number of divisions (or buckets) along the ith dimension. Each value in this matrix represents a fraction of the overall traffic. Thus, the sum of all entries in T is 1.

We also define matrix H as what we call the *hidden distribution*. Each cell in H represents a fraction of traffic generating entities (IP addresses). The attacker specifies how traffic is spread across the divisions of the attack dimension, but for all other dimensions, the hidden dimension is used to distribute attack traffic. The matrices T and H represent the board setup for a single game.

Example 1. *Say there are two dimensions* 1 *and* 2 *(each with 2 divisions). The base and hidden distributions can each be represented as* 2×2 *matrices. Let the base*

المنارة للاستشارات
www.manaraa.com

distribution be

$$T = \begin{bmatrix} 0.1 & 0.3 \\ 0.4 & 0.2 \end{bmatrix}.$$

Note, for example, that the amount of traffic reaching the server from IP addresses that fall in division 1 *of the first dimension (rows) and division* 2 *of the second dimension (columns) is* 0.3 *(*$= T[1,2]$*) fraction of the total traffic reaching the server. Let the hidden distribution be*

$$H = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}.$$

Note, for example, that the number of IP addresses that fall in division 1 *of the first dimension and division* 2 *of the second dimension is* 0.2 *(*$= H[1,2]$*).*

3.2. ATTACKER'S AND DEFENDER'S KNOWLEDGE

We will assume limited knowledge for both the attacker and the defender. The defender can know the typical (non-attack) distribution for each IP address. However, for two reasons the defender cannot apply statistical filtering at this fine-grained level. First, the amount of noise at this fine-grained level is significant. Second, even if the noise was negligible, there is a significant price to be paid for implementing a statistical filter at this level. For example, in the simplest scheme, 2^{32} numbers (one per IP address) must be stored and looked up for each packet. For these reasons, we will assume that the defender uses only aggregate information. In particular, the defender uses the base distribution along one, say the jth dimension.

We will assume that the attacker has knowledge of one dimension, say the ith dimension. What this means is that the attacker knows how much traffic is typically generated by nodes in each division of dimension i, and the attacker knows the set of IP addresses in each *division* of dimension i. In particular, it can use this knowledge to inject traffic with spoofed IP addresses such that the distribution of traffic (or

المنارة للاستشارات

www.manaraa.com

relative traffic) along dimension i does not change from its base distribution.

3.3. INJECTING TRAFFIC

Continuing with Example 1 above, say the attacker has a priori knowledge about the typical historical distribution for dimension 1. Suppose he decides to inject a total α amount of traffic (relative to the base traffic) with distribution $[\beta, 1-\beta]$ along dimension 1 (that is, the first division of dimension 1 gets $\alpha*\beta$ additional traffic, etc). How much traffic is injected in the cell $[1, 1]$, for example? More formally, let i be the dimension that the attacker knows. Let δ_j be the amount of attack traffic that the attacker decides to add to division j of dimension i. Then we assume that the attacker uniformly spreads δ_j across all IP addresses that fall in division j of dimension i. This is a reasonable assumption because the attacker has no knowledge about the other dimensions. Thus, in our example above, cell $[1, 1]$ will get $\alpha * \beta * \frac{0.3}{0.3+0.2}$ amount of additional traffic. After new values for each cell are calculated and the matrix is normalized, we calculate the sensitivity matrix.

3.4. PAYOFF: SENSITIVITY MATRIX

We analyze the game by analyzing the payoff matrix of the game. Let the dimension that the attacker knows be i and let the dimension that the defender uses be j. The attacker's goal is to inject one unit of traffic such that the change in the distribution along dimension j is minimized. The defender wants to detect such a change. The payoff to the defender is the change in the distribution along the jth dimension if a unit amount of attack traffic is introduced using the ith dimension. We call this quantity *sensitivity to the ith dimension along the jth dimension* and denote it as $S[T, H, i, j]$. When T and H are clear from the context, we refer to $S[T, H, i, j]$ as $S[i, j]$. S is called the *sensitivity matrix.* Periodically we will also refer to simply the *sensitivity* of an experiment, in which case we mean the maximum sensitivity value of a sensitivity matrix. This value represents the best payoff for the defender. Since the game is a zero-sum game, the payoff to the attacker is the negation of the payoff to the defender.

It is easy to see that in general the sensitivity function is asymmetric with respect to dimensions. That is, for a given attack A, and dimensions i and j, $S[i, j]$

المنارة للاستشارات

www.manaraa.com

is not necessarily equal to $S[j, i]$.

3.5. EFFECT OF INJECTING ATTACK TRAFFIC

Given the base and hidden matrices, we now show how to analytically compute the effect of injecting attack traffic. As a special case we get how to compute the sensitivity matrix.

Let T be the base distribution and let H be the hidden distribution on n dimensions such that dimension i has d_i divisions. Let α be the amount of traffic to be injected relative to the total amount of traffic represented by the base distribution. Let q be the attacker's dimension and let r be the defender's dimension. Let A be a d_q-dimensional vector representing the attack traffic distribution. (The rth entry in A denotes the fraction of attack traffic that is added to the rth division of dimension q.)

Let $H[1:i_1, 2:i_2, \ldots, n:i_n]$ denote the cell of H whose label along dimension j is i_j, for each $j \in \{1, 2, \ldots, n\}$. Let $H[k:\ell]$ denote an $n-1$ dimensional matrix H' with dimensions $\{1, 2, \ldots, k-1, k+1, \ldots n\}$ such that the $H'[1:i_1, 2:i_2, \ldots, k-1:i_{k-1}, k+1:i_{k+1}:i_n]$ is $H[1:i_1, 2:i_2, \ldots, k-1:i_{k-1}, k:\ell, k+1:i_{k+1}, \ldots, n:i_n]$. Thus, informally speaking, $H[k:\ell]$ denotes a $n-1$ dimensional *slice* of H containing all cells for which the dimension k label is ℓ. Note that $0 \leq i_j < d_i$, for each dimension i in H.

We also define operator $H_{|j}$ (the *projection of H along dimension j*) as a vector of size d_i such that the i-th component is the *sum* of entries in the matrix $H[i:j]$. Next, we define the operator $agg(H)$ which is the set of all n sum vectors of H, where vector i of $agg(H)$ is $H_{|j}$ and has d_i values.

We calculate the fraction of attack traffic that will be added to a cell C labeled $[1:i_1, 2:i_2, \ldots, n:i_n]$:

$$\Delta[1:i_1, 2:i_2, \ldots, n:i_n] = A[i_q]\frac{H[1:i_1, 2:i_2, \ldots, n:i_n]}{agg(H[q:i_q])}$$

Note that the $A[i_q]$ term denotes the fraction of attack traffic added to the i_qth division of dimension q and the fractional term is the number of IP addresses that lie in cell C as a fraction of the number of IP addresses that lie in the i_qth division of dimension q. Note that $agg(\Delta) = 1$.

المنارة للاستشارات
www.manaraa.com

Finally, we can calculate the new base matrix T' that results when the attack traffic is added to the original base matrix T:

$$T' = \frac{T + \alpha\Delta}{1 + \alpha} \tag{1}$$

The new distribution along dimension r is $T'_{|r}$. The *effect of attack A volume α in the qth dimension as observed in the rth dimension* is

$$E(A, \alpha, q, r) = \sum_{1 \leq i \leq d_q} |T_{|r}[i] - T'_{|r}[i]|.$$

The *sensitivity of attack A in the qth dimension as observed in the r dimension* is defined as

$$S(A, q, r) = E(A, 1, q, r).$$

Note that the sensitivity is the effect of a unit attack traffic.

We show below how the effect of of an attack A with an arbitrary amount of traffic is related to the sensitivity of the same attack.

Lemma 2. *For each $\alpha \geq 0$, $E(A, \alpha, q, r) = \frac{2\alpha}{1+\alpha} S(A, q, r)$.*

Proof omitted due to space considerations.

3.6. MODELING NOISE

The base distribution represents a historical pattern developed from a snapshot of traffic, and the traffic injection game is played based on something assumed to resemble this historical pattern, thus we need to model the effect that noise has on the game and calculate the new traffic matrix T' that results when the noise and attack traffic are added to the original base matrix T.

Let N be the noise distribution on n dimensions such that dimension i has d_i divisions. Thus the noise distribution is the same shape as the base and hidden distributions. We will consider the noise as a normal distribution that is independent of the base distribution, and we define ν as the noise level relative to the base traffic volume.

المنارة للاستشارات

www.manaraa.com

Using this definition of noise, we can now calculate the new traffic matrix T' as:

$$T' = \frac{T + \nu N + \alpha\Delta}{1 + \nu + \alpha} \tag{2}$$

If we set $\nu = 0$ then Equation 2 is equivalent to Equation 1.

3.7. METRICS

The sum of the positive differences between the new volume and the base volume when the volumes are projected onto each dimension constitute a sensitivity measurement for a given defense dimension. A table of sensitivity measurements by attack dimension and defender dimension constitutes the $n \times n$ sensitivity matrix. From the sensitivity matrix, we calculate the six metrics described below.

We identified a number of aggregations we want to capture about this game. For a set of sensitivity matrices, we want to know what the max payoff (plus mean, stdev), min payoff (plus mean, stdev), maxmin payoff (plus mean, stdev), and minmax payoff (plus mean, stdev) are across all the cells of the sensitivity matrices. This should allow us to identify the expected payoff under various scenarios.

We have taken six key measurements from the sensitivity matrices generated by the game.

- *BIMA:* Best independent move for the attacker. This is the max-minimum value for any row in the sensitivity matrix.
- *BIMD:* Best independent move for the defender. This is the min-maximum value for any column in the sensitivity matrix.
- *AOS:* Average overall sensitivity. Simple average of all values in the sensitivity matrix.
- *AOP:* Average overall payoff. This is the average of the value at the intersection of the sensitivity matrix of the BIMA row and the BIMD column.
- *DDA:* Delta defection to best move for the attacker. Starting at the intersection of the BIMA row and BIMD column in the sensitivity matrix, find the lowest value in that column that the attacker could choose if he changed moves.

المنارة للاستشارات

www.manaraa.com

- *DDD:* Delta defection to best move for the defender. Starting at the intersection of the BIMA row and BIMD column in the sensitivity matrix, find the highest value in that row that the defender could choose if he changed moves.

3.8. ATTACK TYPES

We have analyzed four kinds of attacker:

- *Random:* The ratio of attack traffic for each division of the attack dimension is a randomly chosen normalized distribution.
- *Base:* Attack traffic is spread so that it matches the distribution for divisions in the base traffic distribution for the attack dimension.
- *Uniform:* Attack traffic is spread evenly amongst the divisions in the attack dimension. For example, if there are 10 divisions, each gets 10% of the attack traffic.
- *Loaded:* This kind of attacker directs all of the attack traffic at division 0 of the attack dimension.

For each of the four attacker strategies, our analysis consisted of generating many base and hidden distributions, and for each of these an attack distribution and noise distribution are generated many times. For each of these, a sensitivity matrix was generated so that we could analyze all possible moves for attacker and defender. The results were aggregated and the average and standard deviation reported by the program.

3.9. EQUILIBRIUM AND DISTRIBUTION DIFFERENCE ANALYSIS

One important aspect of the the traffic injection game to discuss is the equilibrium of the game. Assuming full information and no noise given that there are n dimensions or actions for each party then it is easy to go exhaustively over all n^2 possibilities to check for pure Nash Equilibrium (NE) in polynomial time. Mixed NE can be computed exactly using linear programming. However our goal here is not to solve for NE since that assumes full information and in reality the attacker and

المنارة للاستشارات
www.manaraa.com

defender may be unaware of H, T, etc. We are after an understanding in the imperfect information model, so we study how the sensitivity varies with the *difference* between the distributions in our model. There are two basic types of distributional differences:

Interdimensional distributional differences occur between distributions along two dimensions of the base distribution or they occur between distributions along two dimensions of the hidden distribution. Intuitively speaking, if there is a large interdimensional difference between the attacker's and defender's dimensions, then the sensitivity is high.

Base-Hidden distributional differences occur between two distributions—a projection along a dimension of the base distribution and a projection along a dimension of the hidden distribution.

Because of the complex relationship between the base and the hidden distributions, a complete analysis of these differences and the equilibrium of the game are outside the scope of this paper (and is left as future work). However, we present an analysis of a special case of base-hidden difference. In particular, we study the change in sensitivity in the following case. We are given a base distribution $\hat{T}$. The attacker's dimension is 1 and the defender's dimension is 2. (Note that this is not a restriction because we can rename dimensions.) We want to find hidden distributions H such that the sensitivity is maximized and the following hold for each dimension $j \neq 2$:

$$H_{|j} = \hat{T}_{|j}.$$

That is, the base and hidden distributions match in all dimensions except the defender's dimension.

This problem can be expressed as the following optimization problem:

$$\max S[\hat{T}, H, 1, 2]$$
$$\text{s.t.}$$
$$H_{|j} = \hat{T}_{|j}, \quad \forall j = 1, 3, 4, \ldots, n.$$

Formally, a *distribution with k divisions* is a k-dimensional vector $[x_1, x_2, \ldots, x_k]$ such that each $0 \leq x_i \leq 1$ and $\sum_{i=1}^{k} x_i = 1$. The difference between two distributions

المنارة للاستشارات
www.manaraa.com

$X = [x_1, x_2, \ldots, x_k]$ and $Y = [y_1, y_2, \ldots, y_k]$ (each with k divisions) is

$$\sum_{i=1}^{k} |x_i - y_i|.$$

We show in the results section that the optimal solution to this problem increases linearly with the difference between $T_{|2}$ and $H_{|2}$. Intuitively, it is safest for the attacker not to attack, if his goal is not to get detected. As the amount of attack traffic increases (as shown for the special case above), the higher the chance of the attacker being detected by the defender.

المنارة للاستشارات
www.manaraa.com

4. TIGGER: SOFTWARE

Traffic Injection Game Graphical Engine for Results (TIGGER) is a software tool developed for performing reproducible simulation and analysis required by this research. The parameters that are recognized by the program consist of the following:

- Dimensions: set of comma-delimited numbers indicating how many divisions make up each dimension. Example: "2,2", "10,10,10", "5,2,8*". Suffixing a number with an asterisk directs the values in that dimension to be generated using a power law distribution. Otherwise, the values are a randomly chosen, normalized distribution.
- Attacker type: Base, Random, Uniform, or Loaded.
- Noise Level: decimal number specifying the noise ratio relative to the base traffic. The base traffic level is always considered to be 1.0.
- Attack Traffic: decimal number specifying the ratio of attack traffic volume relative to the amount of base traffic.
- Random Seed: integer seed value (for reproducibility).

The program also supports iteration by repeating experiments for various values for noise level and attack traffic volume.

المنارة للاستشارات
www.manaraa.com

5. RESULTS

Based on the results of the two previous examples, we can design a statistical filter that analyzes traffic patterns, identifies the division within the observed dimension that represents the most suspicious group of traffic, and recommend that group of traffic for more rigorous, albeit time-consuming, verification techniques. It is not necessary to limit the functioning of such a filter to just the most suspicious group of traffic. The filter could also be designed to send any group of traffic whose traffic increases above a certain threshold to be scrutinized more carefully.

The base traffic pattern in a real-life scenario would need to account for per hour patterns, since the dynamics of many dimensions from which a defender could observe will change based on the time of day as found in our research.

If the statistical filter were combined with a feedback loop that processes emerging traffic patterns back into the expected base traffic values, the benefit of the filter might even be able to be improved.

The rest of this section describe the results that our implementation of the traffic injection game produced, using various attacker types, noise levels and attack traffic volumes. This allows us to analyze the effect of such parameters on the 6 metrics of which we have been discussing.

5.1. VARIATION WITH ATTACK TRAFFIC AND NOISE

The best independent move for the attacker (BIMA) is a maximin value that identifies the best that the attacker could do if the defender chose the worst dimension to watch based on the attacker's move. If the attacker knows the dimension that the defender will observe, then the attacker is always going to choose the dimension that generates the minimum sensitivity value for the defense dimension. This means that the defender's suspicion was minimally heightened.

Figure 5.1 shows the effect of varying the attack traffic ratio given a fixed level of noise (0.3), while Figure 5.2 shows how the BIMA is affected by fixing the attack

المنارة للاستشارات
www.manaraa.com

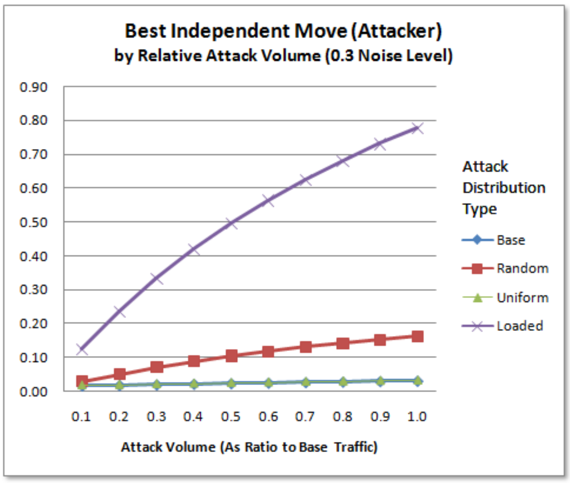


Figure 5.1. Plot of best independent move for the attacker versus attack volume on a 10x10x10 matrix with 0.3 noise level.

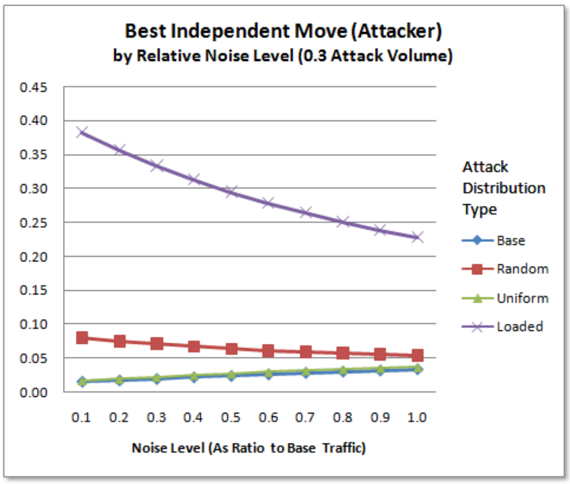


Figure 5.2. Plot of best independent move for the attacker versus noise level on a 10x10x10 matrix with 0.3 attack volume.

traffic at 0.3 and varying the noise level. As we see, the best strategy for the attacker is observed when the attack traffic is distributed randomly or when the attacker is aware of and employs a distribution that matches the base distribution for the attack dimension. The random strategy does as well because of the stochastic nature of the generated distributions. The worst performance comes when the attacker blindly,

المنارة للاستشارات

www.manaraa.com

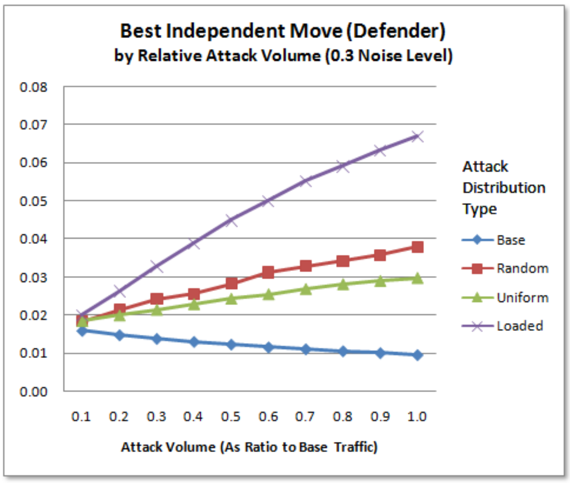


Figure 5.3. Plot of best independent move for the attacker versus attack volume on a 10x10x10 matrix with 0.3 noise level.

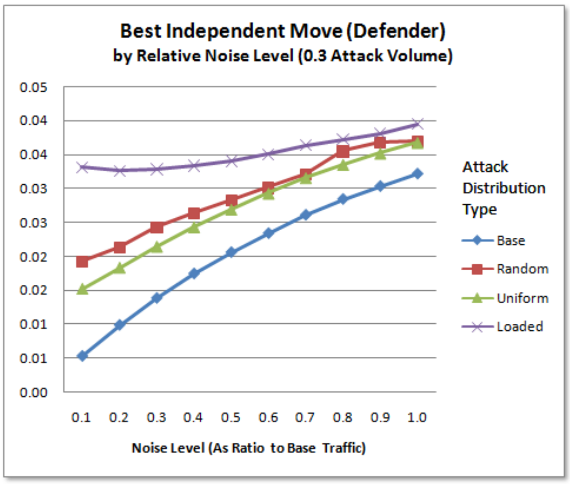


Figure 5.4. Plot of best independent move for the attacker versus noise level on a 10x10x10 matrix with 0.3 attack volume.

evenly distributes traffic or blindly applies all its resources at a single distribution, illustrating that a powerful, wise attacker performs better.

Figure 5.1 also illustrates the fact that as the attack traffic increases, the uniform and loaded strategies fare worse as the attack traffic becomes more and more exposed relative to the base traffic and noise, while the attacker's best move gets no worse

المنارة للاستشارات
www.manaraa.com

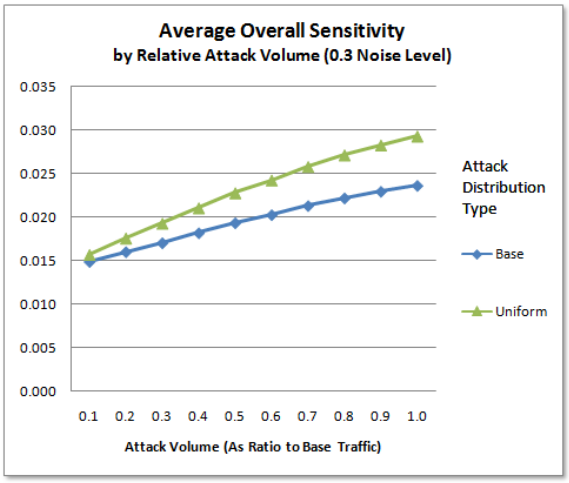


Figure 5.5. Plot of average overall sensitivity versus attack volume on a 10x10x10 matrix with 0.3 noise level.

by increasing his attack volume. Figure 5.2 shows that as the noise level increases relative to the base and attack traffic, the poorer strategies are hidden a little better. This would support the idea that as observed traffic deviates more and more from established patterns, even bad attack strategies become harder to detect.

The best independent move for the defender is analyzed similarly, however in this case the values are calculated based on the best that the defender can do given that the attacker can choose the dimension that presents the worst max value for the defender, or a minimax value. As can be seen in Figure 5.3 and Figure 5.4, the strategies that worked well from the attacker's point of view also present the biggest problems for the defender, and vice-versa.

The average overall sensitivity (AOS) values, as seen in Figure 5.5 and Figure 5.6, is a metric of the best potential payoff that the defender could achieve in any dimension for a particular experiment. The base and uniform attacker types perform with this metric almost identically, with the figures isolating only these two. As the noise level and the attack volume increase, so does the average overall sensitivity measurement. This also shows that as an attacker, using knowledge about the expected distribution of traffic in the dimension being attacked is a powerful tool for hiding attack traffic.

The average overall payoff (AOP) value, as seen in Figure 5.7 and Figure 5.8

المنارة للاستشارات

www.manaraa.com

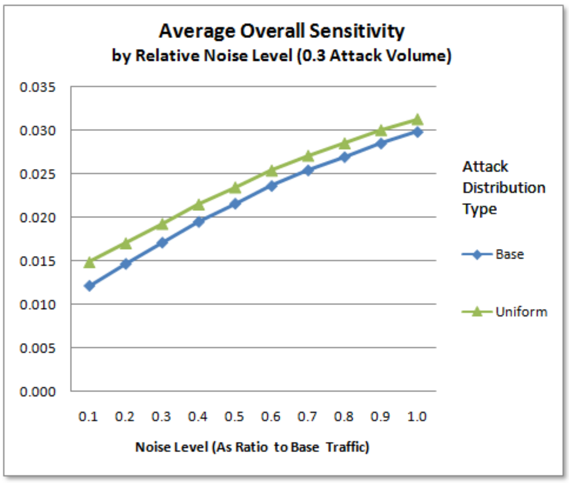


Figure 5.6. Plot of average overall sensitivity versus noise level on a 10x10x10 matrix with 0.3 attack volume.

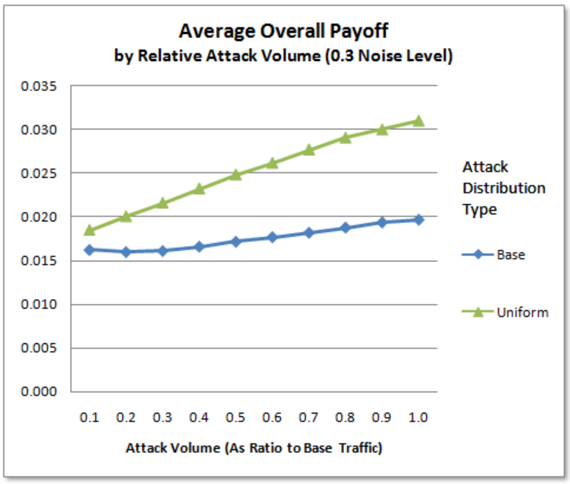


Figure 5.7. Plot of average overall payoff versus attack volume on a 10x10x10 matrix with 0.3 noise level.

shows what the average of the expected payoff would be, given that the attacker chooses the best independent move for the attacker, and the defender chooses the best independent move for the defender. The value in the sensitivity matrix at this junction is the overall payoff, and the AOP is the average value as measured in each experiment. As can be seen in the figures, the pattern followed is similar to the

المنارة للاستشارات
www.manaraa.com

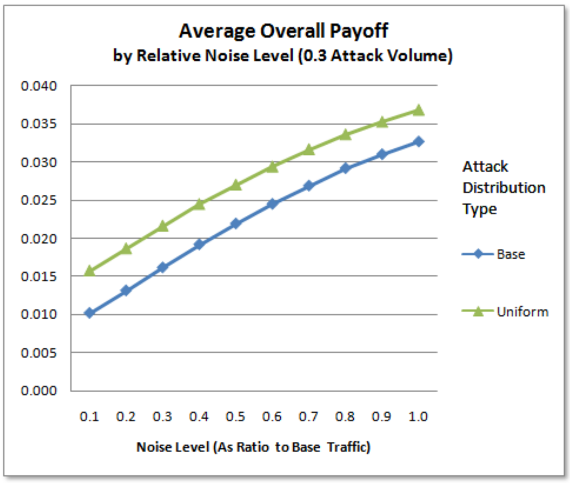


Figure 5.8. Plot of average overall payoff versus noise level on a 10x10x10 matrix with 0.3 attack volume.

sensitivity for this game. As the noise level and the attack volume increase, so does the average overall payoff measurement. An attacker wishing to optimize this metric would be best served by a strategy taking advantage of knowledge about the expected distribution of traffic in the dimension being attacked.

Another conclusion illustrated by these results is that a defender that has knowledge of certain dimensions within the system that the attacker does not is able to better expose that an attack is occurring and from where that attack is coming. This fact is shown by examining the loaded attacker type. In this type of attack, the attacker is applying attack traffic to a single division within its attack dimension, and regardless which dimension the defender is watching, the attacker exposes its traffic the most of any of the attack strategies we examined.

5.2. DISTRIBUTION DIFFERENCE ANALYSIS-INTERDIMENSIONAL

As discussed above in Section 3.9, interesting things happen when we vary the distance between distributions. There are several components we may consider when analyzing the data in this way. First, we can vary the distance between the aggregate distributions for the divisions of two dimensions of the base distribution, fixing the attack distribution. We can also fix the base and attack distributions, and vary the

www.manaraa.com

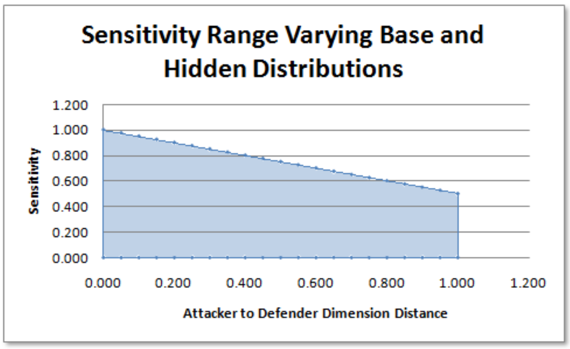


Figure 5.9. Plot of sensitivity values as the distance between the attacker and defender dimensions vary.

distance between the base and hidden distributions.

First, we consider a game consisting of symmetric randomly chosen distributions, fix the attacker type as a Base attacker, and then analyze the sensitivity matrices that result. As shown in Figure 5.9, the sensitivity grows as the distributions used by the attacker and defender approach zero. This means that the defender has chosen a dimension whose distribution perfectly isolates the attack traffic. As the two distributions diverge, the attack traffic becomes more hidden from the defender.

5.3. DISTRIBUTION DIFFERENCE ANALYSIS-BASE TO HIDDEN

When the base and hidden distributions coincide, we say the distance between the two distributions is zero. In this case, and when the attacker is allowed awareness of and he emulates the base distribution for the attack dimension, then the sensitivity matrix contains all zeroes. In other words, the attack is perfectly hidden from the defender. On the other hand, when the distance between the base and hidden distributions is the greatest, the sensitivity is maximal. These observations are illustrated in Figure 5.10 at the bottom left and top right edges of the shaded areas.

One interesting aspect of our research is the fact that even when the attacker mimics the base distribution in the attack dimension, there are still many hidden distributions that result in an aggregate distribution in the defender's chosen dimension that match the base distribution in that dimension.

We next further examine what happens when we fix the base distribution and

المنارة للاستشارات
www.manaraa.com

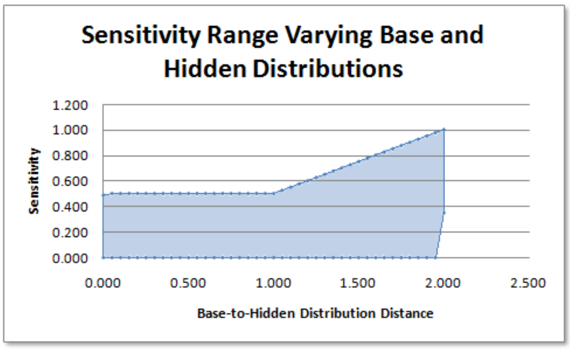


Figure 5.10. Plot of max sensitivity values varying the distance between the attacker and defender dimension aggregate values.

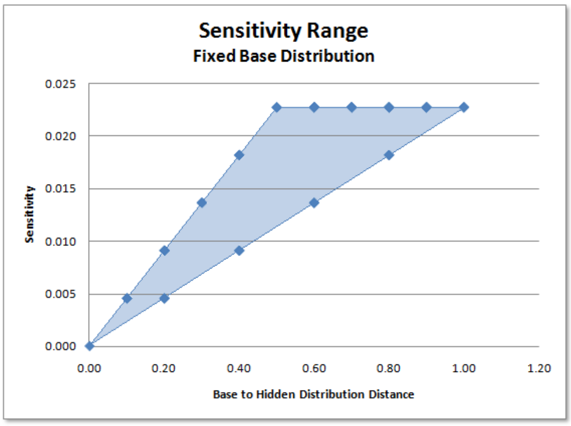


Figure 5.11. Plot of sensitivity values for a fixed base distribution varying the hidden distribution.

show the effect of changing the hidden distribution. As shown in Figure 5.11, varying the hidden distribution so that the distance between it and the fixed base distribution grows causes the sensitivity to increase.

www.manaraa.com

6. CONCLUSION

As we have shown, our research indicates that there is potential for using statistical methods such as ours for creating defense mechanisms that can detect a DDoS attack, given that favorable signal-to-noise ratio exists. Our work also shows that an attacker is more exposed as the frequency of attack traffic increases, but that an attacker that has a priori knowledge of the expected traffic volume for the dimension and divisions employed in the attack has the best ability to hide attack traffic when such statistical methods are used by the defender. Future work seeks to show similar results when real world distributions such as those modeled in a power law are integrated into the sensitivity matrix and employed by the attacker and defender.

المنارة للاستشارات
www.manaraa.com

BIBLIOGRAPHY

[1] Agah, A., Asadi, M., and Das, S. K. Prevention of dos attack in sensor networks using repeated game theory. In *ICWN* (2006), pp. 29–36.

[2] Cominetti, R., Correa, J. R., and Moses, N. E. S. Network games with atomic players. In *ICALP (1)* (2006), pp. 525–536.

[3] Douligeris, C., and Mitrokotsa, A. Ddos attacks and defense mechanisms: classification and state-of-the-art. *Comput. Networks 44*, 5 (2004), 643–666.

[4] Fanglu Guo, Jiawu Chen, T.-c. C. Spoof detection for preventing dos attacks against dns servers. In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)* (2006), p. 37.

[5] Guo, F., Chen, J., and cker Chiueh, T. Spoof detection for preventing dos attacks against dns servers. *icdcs 0* (2006), 37.

[6] Roughgarden, T., and Tardos, E. How bad is selfish routing? In *IEEE Symposium on Foundations of Computer Science* (2000), pp. 93–102.

[7] S., A. S. G. D. L. C. R. S. S. An efficient filter for denial-of-service bandwidth attacks. In *Proc. IEEE GLOBECOM '03.* (Dec. 2003), vol. 3, pp. 1353 – 1357.

[8] Schuba, C. L., Krsul, I. V., Kuhn, M. G., Spafford, E. H., Sundaram, A., and Zamboni, D. Analysis of a denial of service attack on TCP. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy* (May 1997), IEEE Computer Society, IEEE Computer Society Press, pp. 208–223.

[9] von Stengel, B. Computing equilibria for two-person games.

www.manaraa.com

II. PREPROCESSING DNS LOG DATA FOR EFFECTIVE DATA MINING

Domain Name Service (DNS) servers provide a critical function in directing Internet traffic. Defending these services from bandwidth attacks is assisted by the ability to effectively mine DNS log data for information that can be used to identify strategies employed by attackers and to develop tools to thwart such strategies. One problem that arises when approaching the data mining task occurs during the data preprocessing phase, when the data captured does not continuously cover the observed time span. These gaps in the data can cloud results and must be filled by estimating the traffic as viewed from any indexed view of the data. We demonstrate a useful method for estimating the values for missing portions of DNS log data and illustrate several queries performed on the data before and after the data cleaning process. This method would be suitable for use with a variety of datasets containing time series values where certain portions are missing.

المنارة للاستشارات
www.manaraa.com

1. INTRODUCTION

The Domain Name System (DNS) is an example of a system that is highly susceptible to DoS type attacks. Numerous examples have been documented, including [5]. When analyzing such systems, one technique involves the capture of log data and the use of data mining techniques to discover trends and other knowledge that might lead to the development of tools and techniques for preventing such attacks [4]. During the data mining process, care must be taken to verify the quality of the source data. In the case of DNS log data, one deficiency that arises is the presence of gaps, or “holes”, in the data where for one reason or another the DNS server failed to log the activity during a period of time. Due to the difficulty of capturing and obtaining a set of log data, it is often not possible to simply go get another set of data, and so the data must be cleaned before it is used.

There are a number of questions asked about DNS log data in such analysis, or of time series data in general. For example, we would like to characterize a typical traffic frequency pattern for a set of DNS servers. If there are holes in the source data, however, the results of such an exercise would not be useful because they would not represent reality. Basing analysis on such faulty aggregation from incomplete source data would skew those results and could lead to incorrect conclusions. In fact, in our analysis efforts we noticed the missing data only after trying to draw conclusions and noticing odd, counter-intuitive trends in the data. This warranted further investigation, which lead to detecting the data insufficiency situation. The dataset is so large, and the holes in the data so disjointed, that when viewing the aggregated results it was not easy to notice.

Upon discovering such a problem, several options are available. The first option would be to simply abandon the data and seek another source. However, the process of capturing the data is time-consuming and resource-intensive for the DNS servers, the data set is large and difficult to store, copy and organize for processing, and the sensitivity of the data requires access to be regulated. As a result we needed to find

المنارة للاستشارات

www.manaraa.com

a way to adjust the data.

A second option was to manufacture new DNS request records and add them to the data source, marking them as fake requests to distinguish them from the real ones. From there we would compute our aggregate values from both the real and fake records. But this would add to the amount of data storage space required for the dataset and would ergo increase the time needed to process the data to compute the aggregates.

The method we illustrate in this paper involves aggregating the data first, and then adjusting those values, providing a balance between storage, speed in getting answers to queries, and better accuracy because of the filled holes.

المنارة للاستشارات
www.manaraa.com

2. DNS LOG DATASET

The dataset we had to work with was obtained from a large company central to the management of the DNS network. The dataset has data collected at 26 root servers over a 10 day period in January 2004. For each server, we have log data that includes all DNS requests made from client DNS servers. The log files consisted of over 400 gigabytes of raw DNS log data.

The log files contain many different kinds of records[1][2], including simple header records that indicate what version of software is running on the DNS server. Almost half of the record types do not involve a response that contains an IP address. As part of the data cleaning and integration stage [3], a utility was developed to scan the log files and segregate them into groups of records using regular expressions. The first group were the header records, and these were just thrown away. The next group includes any records that return an IP address, which is the primary group of focus for this analysis. The third group was any record that looked to be well-formed, but simply did not return an IP address. There are many record types that fit this category. These were saved for potential examination and analysis later. The last group were classified as error records. These include records that were apparently malformed to the DNS server, and the record indicates the problem diagnosis. These records were also set aside for later potential examination.

As the records were segregated, output files generated were in the format of bulk-insert files for SQL Server based on a database model that captures all of the relevant data from each request necessary for later analysis. Problems with the initial data model became apparent after importing the data into the database. The sheer size of the data presented a number of challenges to the available disk space, memory, and CPU power. After a few attempts, a workable plan was developed that included a data model to use during the import phase, with modifications to be done to the data model once all the data had been imported. This resulted in a more efficient import process.

المنارة للاستشارات
www.manaraa.com

The database model consisted of a single clustered index on the client's IP address in anticipation of this being the most heavily queried-against column. Additional indices were not added prior to the bulk insert phase, as this would slow the import process. The quickest way to add an additional, non-clustered index was estimated to be to add the index after the data had all been imported. Future work may benefit from indexed views available in SQL Server 2000 (Enterprise Edition) or SQL Server 2005. These indexed views allow storage of aggregate values within the index and thus avoid a complete table scan when scanning the index will suffice.

The table containing the log entries was modeled with the columns in Table 2.1.

LogEntryID	bigint	A unique 64-bit identity column.
EntryTime	datetime	The date and time that the request occurred
IPSource	int	The client's IP address, converted to a four-byte integer value
IPServer	int	The DNS server's IP address, converted to a four-byte integer value
TTL	int	The time-to-live value, indicating how many seconds in the future the answer should be considered valid
Type	char(4)	The first four bytes of the request type as it came in on the request
URL	char(128)	The domain name requested by the client

Table 2.1. Columns in LogEntries table capturing pertinent facts from DNS log files.

Originally, domain names were modeled to be stored in a separate table, however, looking up the domain name to determine the foreign key value to insert into the record was taking a long time. Instead, the URL column is used to store the domain name requested, and at a later time, we may normalize the data and replace the URL with a foreign key to a table of domain names. This will likely result in an overall savings in space, but this is actually overshadowed by the amount of space it would take to actually make the changes later (see discussion above about changing the clustered index after the data has been imported).

المنارة للاستشارات
www.manaraa.com

3. DATA SUFFICIENCY

The initial analysis of the server traffic yielded some interesting results, not in the actual patterns that were discovered, but in the concerns that were raised regarding the quality and sufficiency of the available data. As can be seen in Figure 3.1 there are "holes" in the dataset, which are hours of the data collection period for which we have no data or partial data. The figure shows the minutes of data available for each hour. These holes in the data present a challenge to the data mining exercise because incomplete data could create the appearance of false patterns or mask useful patterns.

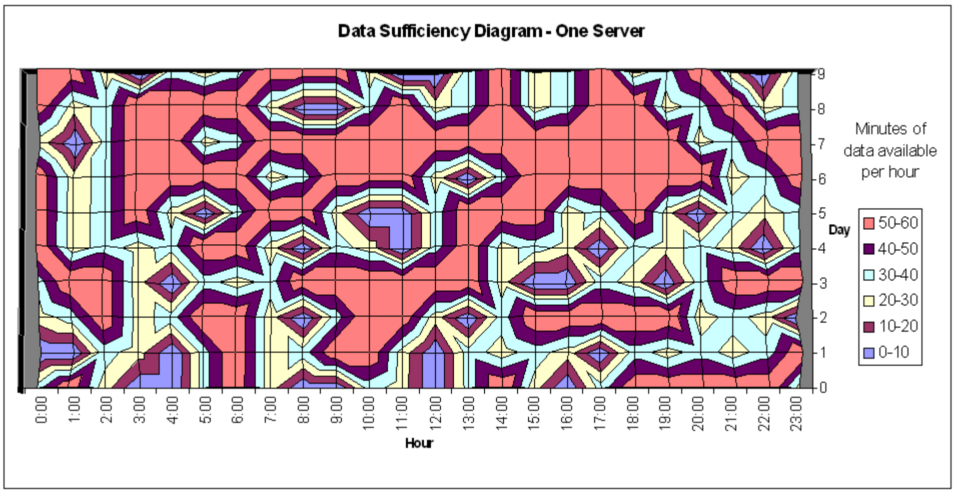


Figure 3.1. Data sufficiency diagram for one server showing the holes in the dataset. The orange regions at the top level (plateau) of the graph shows contiguous days and hours within a day where data is complete. Data points below the top level indicate hours of days with partial data. The blue areas near the bottom level (floor) of the graph indicate hours of days with no data whatsoever.

There are two reasons why we might perceive a hole in the server data, either (a) the server became unavailable due to power failure, communication failure, or

المنارة للاستشارات
www.manaraa.com

for some other reason, or (b) the server just appeared to be unavailable because we simply do not have the logged data during the time period.

Any hole is important because it speaks to the issue of data sufficiency. If the reason is (a) then we have all of the data that was available, but the data isn't useful for our purposes because it does not represent a "normal" traffic pattern. If the reason is (b) then we obviously have a data sufficiency concern, although we might be able to massage the data to get a suitable dataset anyway.

The next question is whether or not it is possible to identify which reason is most likely in each case, and then to develop a process to fill the holes so the resulting dataset is suitable.

One method of explaining a hole would be to examine the server traffic, specifically the hour of data following a hole. Within the algorithm used by clients to load balance across DNS servers, sometimes referred to as "explore and exploit", when a DNS server becomes unavailable, clients and client name servers bound to that DNS server will start utilizing other DNS servers instead. The unavailable DNS server will receive a heavy penalty for not responding, but will still be polled periodically. For this reason, the log data should show that when the DNS server becomes available once more, many clients will have begun relying on other DNS servers, but assuming the DNS server comes back online with the same responsiveness it had before, eventually it should begin to receive traffic volumes consistent with the pattern established prior to it becoming unavailable. Thus, if a hole was due to a server becoming unavailable, then the time period following the hole should show a notable reduction in traffic from what a reasonable projection would have predicted. If this is the case, then we should discard the data from this server as not a candidate for analysis.

Another method would be to examine client traffic, specifically the traffic during the time of the hole. If the hole were caused by reason (a) then the aggregate client traffic across all servers to which it is bound should remain consistent as the client simply looks to other DNS servers to fulfill its needs. However, if the hole were due to reason (b) then there should be a drop in traffic to the server with the hole while the traffic to other servers to which the client is bound should remain statistically consistent with their previous pattern.

Next, once data from servers with holes of type (a) have been eliminated from

المنارة للاستشارات

www.manaraa.com

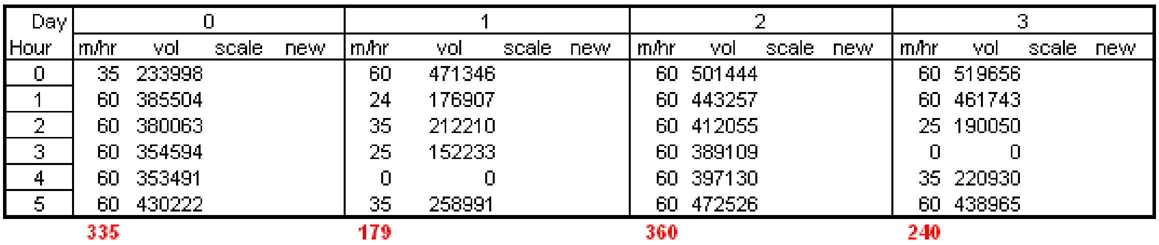
consideration, the next issue is how to fill the holes so that the dataset represents a projection of traffic that follows the pattern set for other days, for other hours of the same day, and for other servers during the time period of the hole.

المنارة للاستشارات
www.manaraa.com

4. FILLING HOLES IN SERVER DATA

The algorithm used for filling the holes in the server data requires several steps. The first step is to count all traffic that each server received for each day and hour of the analysis period. For our data, this amounted to 10 days of data.

The next step was to identify which hours provide complete data, which hours are missing, and for hours with partial data, how many minutes of data are present. The method of data capture made this relatively easy. The log files for each server provide a continuous record of all requests while the log file was being created. So each log file was examined and the date-time stamp for the first and last entry was recorded. Once these start and end date-time values were sorted and examined, a value from zero to 60 was assigned to each server-day-hour that represents the number of minutes of data available.

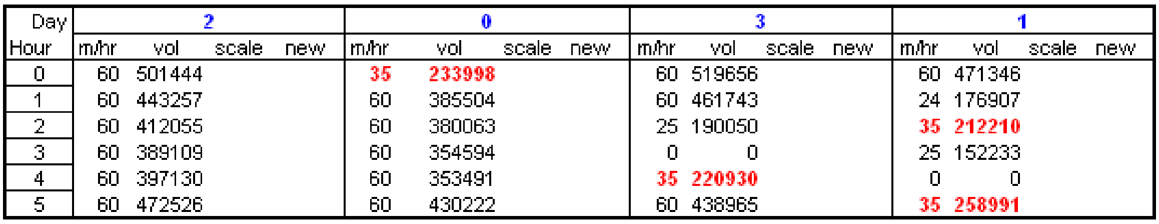


Day	0				1				2				3			
Hour	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new
0	35	233998			60	471346			60	501444			60	519656		
1	60	385504			24	176907			60	443257			60	461743		
2	60	380063			35	212210			60	412055			25	190050		
3	60	354594			25	152233			60	389109			0	0		
4	60	353491			0	0			60	397130			35	220930		
5	60	430222			35	258991			60	472526			60	438965		
	335				179				360				240			

Figure 4.1. First step of analysis showing a partial set of data for one server. In red, the sum of minutes of data for each day are listed.

At this point, we have a record (Figure 4.1) of the original data (count of requests) and a measurement of the reliability of each data point, measured in minutes of data available per server-day-hour. The next step was to sort the days by the overall reliability of the day relative to the rest of the days. In this way, we process days from most reliable to least reliable. Our interpolation method in a future step will use the pattern established by more reliable days to compute an adjusted value.

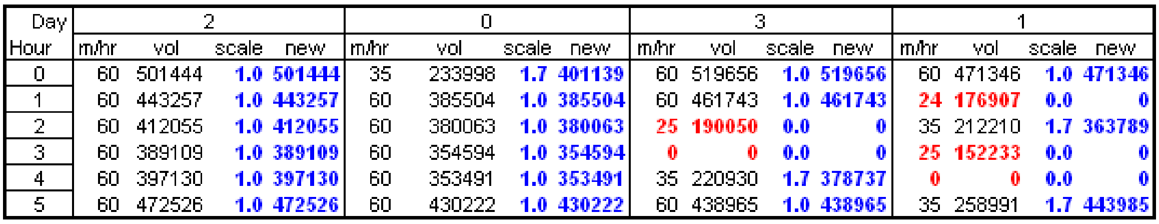
المنارة للاستشارات
www.manaraa.com



Day	2				0				3				1			
Hour	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new
0	60	501444			35	233998			60	519656			60	471346		
1	60	443257			60	385504			60	461743			24	176907		
2	60	412055			60	380063			25	190050			35	212210		
3	60	389109			60	354594			0	0			25	152233		
4	60	397130			60	353491			35	220930			0	0		
5	60	472526			60	430222			60	438965			35	258991		

Figure 4.2. Second step of analysis showing days resorted in order by reliability. In blue are the original day numbers. In red are the hours for each day with partial data.

The next step was to calculate a scaling factor from zero to one for each server-day-hour, and then compute an adjusted traffic value. The scaling factor was calculated as the number of minutes of data available divided by 60. For complete days, this results in a scaling factor of 1.0. For missing hours, the scaling factor is 0.0. One optimization that was applied to the algorithm that is designed to improve accuracy for "light" clients was to consider any server-day-hour with less than 30 minutes of data available as unreliable and so the scaling factor for these data points was set to 0.0 as well. Once the scaling factor was determined, the adjusted traffic value was simply computed as the scaling factor times the original traffic value.



Day	2				0				3				1			
Hour	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new
0	60	501444	1.0	501444	35	233998	1.7	401139	60	519656	1.0	519656	60	471346	1.0	471346
1	60	443257	1.0	443257	60	385504	1.0	385504	60	461743	1.0	461743	24	176907	0.0	0
2	60	412055	1.0	412055	60	380063	1.0	380063	25	190050	0.0	0	35	212210	1.7	363789
3	60	389109	1.0	389109	60	354594	1.0	354594	0	0	0.0	0	25	152233	0.0	0
4	60	397130	1.0	397130	60	353491	1.0	353491	35	220930	1.7	378737	0	0	0.0	0
5	60	472526	1.0	472526	60	430222	1.0	430222	60	438965	1.0	438965	35	258991	1.7	443985

Figure 4.3. Third step of analysis showing scaling factors and adjusted volumes for partial days in blue. Hours with no log data at all are shown in red.

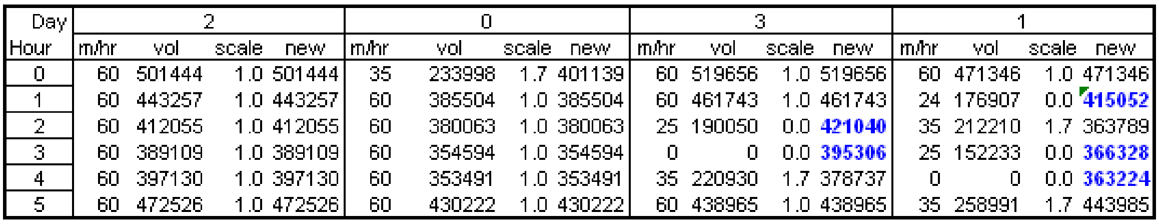
Processing each day in order from most reliable to least reliable as described above, the next step is to fill in the holes. Remember that we consider a hole as any server-day-hour for which there was less than 30 minutes of data available. For the first (most reliable) day, we calculate the adjusted traffic value by simply interpolating

المنارة للاستشارات
www.manaraa.com

between known values. If the first (or last) hour of a day was missing, the first (or last) available adjusted value present was used.

Once the first day has been repaired by computing an adjusted traffic volume value, we moved on to successive less reliable days. For holes in each day, we calculated new volumes as the sum of all non-missing hours this day times the sum of values for this hour for days already processed divided by the sum of all non-missing hours for days already processed.

$$
\begin{aligned}
Vol_{NMHT} =& \text{Sum of volumes of non-missing} \\
& \text{hours this day} && (1) \\
Vol_{AdjDAP} =& \text{Sum of adjusted volumes for days} \\
& \text{already processed for this server} && (2) \\
Vol_{AdjNMHT} =& \text{Sum of adjusted volumes in non-missing} \\
& \text{hours for days already processed} && (3) \\
Vol_{Adj} =& \frac{Vol_{NMHT} * Vol_{AdjDAP}}{Vol_{AdjNMHT}} && (4)
\end{aligned}
$$



Day	2				0				3				1			
Hour	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new	m/hr	vol	scale	new
0	60	501444	1.0	501444	35	233998	1.7	401139	60	519656	1.0	519656	60	471346	1.0	471346
1	60	443257	1.0	443257	60	385504	1.0	385504	60	461743	1.0	461743	24	176907	0.0	**415052**
2	60	412055	1.0	412055	60	380063	1.0	380063	25	190050	0.0	**421040**	35	212210	1.7	363789
3	60	389109	1.0	389109	60	354594	1.0	354594	0	0	0.0	**395306**	25	152233	0.0	**366328**
4	60	397130	1.0	397130	60	353491	1.0	353491	35	220930	1.7	378737	0	0	0.0	**363224**
5	60	472526	1.0	472526	60	430222	1.0	430222	60	438965	1.0	438965	35	258991	1.7	443985

Figure 4.4. Fourth step of analysis showing interpolated values for missing hours in blue.

Alternate methods of repairing the first day could be explored, such as interpolating from multiple data points, or using an alternate method for missing first/last hours of the day, or by processing the first day again after all other days have been processed using the same method for the other days based on adjusted values on

www.manaraa.com

those other days. The improvement in quality was estimated to be minimal, and so for simplicity was omitted.

At this point, the data represents a complete picture of the nature of the traffic for the server with the holes filled in. As a pleasant surprise, the graph of the adjusted traffic volume values has fewer and less severe ripples.

This completes the analysis performed on the server traffic, but this data is important for adjusting the client traffic and so had to be completed first. Since clients are predominantly tied to a particular server (or set of servers), a hole in the server data would generally correspond directly with holes in the client data. Another way to say this is to say that if a client is communicating with a server and we are missing data for the server, then it is reasonable to assume that some of the missing traffic came from that client and so its traffic value will need to be adjusted in proportion to the adjustment made to the server traffic.

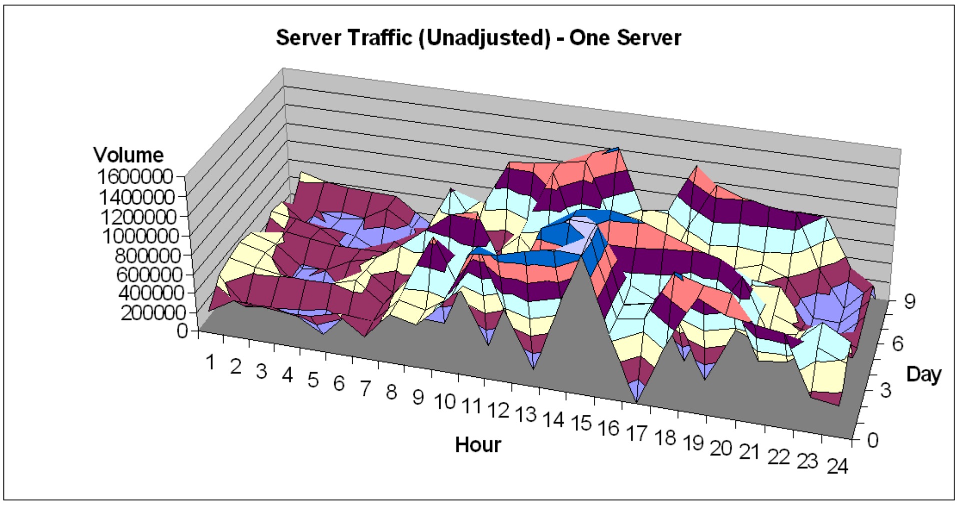


Figure 4.5. Server traffic diagram showing reported traffic (unadjusted) for one server. Note how uneven the data appears.

Figure 4.5 shows a graph of the original traffic from the available log files for one server. Figure 4.6 shows a graph of the same server, after the data has been adjusted to allow for the holes. As you can see, the holes appear to be filled with traffic

المنارة للاستشارات
www.manaraa.com

consistent with what one might expect to have received during these periods. As can be seen, since the formula adjusts for the traffic pattern for a day in general, adjusted based on the observed volume for a specific day, even on days with lower traffic, the pattern of traffic is preserved. This is illustrated in Figure 4.6 by the valley running down the center that represents lower traffic days between the relatively higher traffic days.

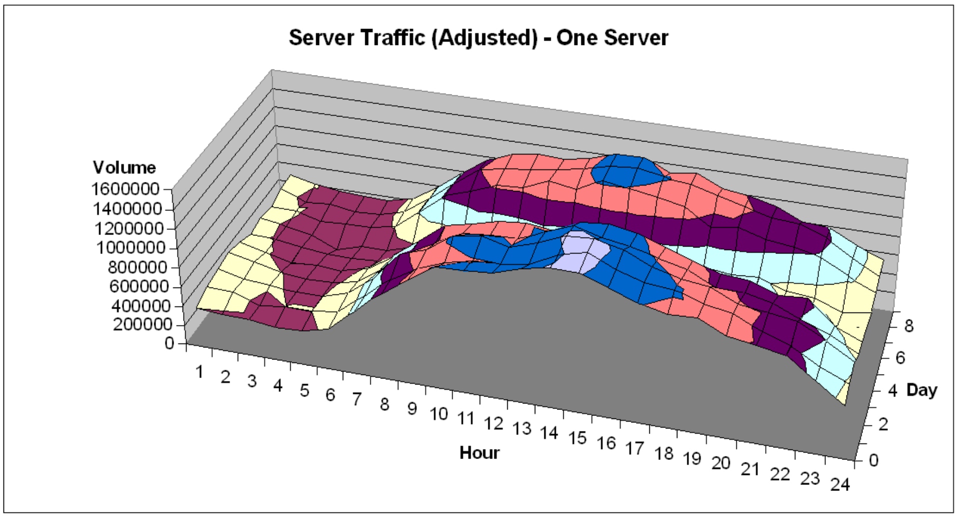


Figure 4.6. Server traffic diagram showing reported traffic for one server adjusted to repair the holes. Note how now the data looks even and smooth.

www.manaraa.com

5. DEMONSTRATING USEFULNESS OF CLEANED DATASET

Once the processing of the original log data had been completed, the result was a set of robust indices that could be used to perform analysis from a variety of perspectives.

5.1. CLIENT POLYGAMY ANALYSIS

An interesting feature of client behavior is found in the number of servers that a client utilizes. When a client relies on one (or a few) DNS servers, we say it is relatively monogamous. The more DNS servers utilized by a client, the more polygamous it is.

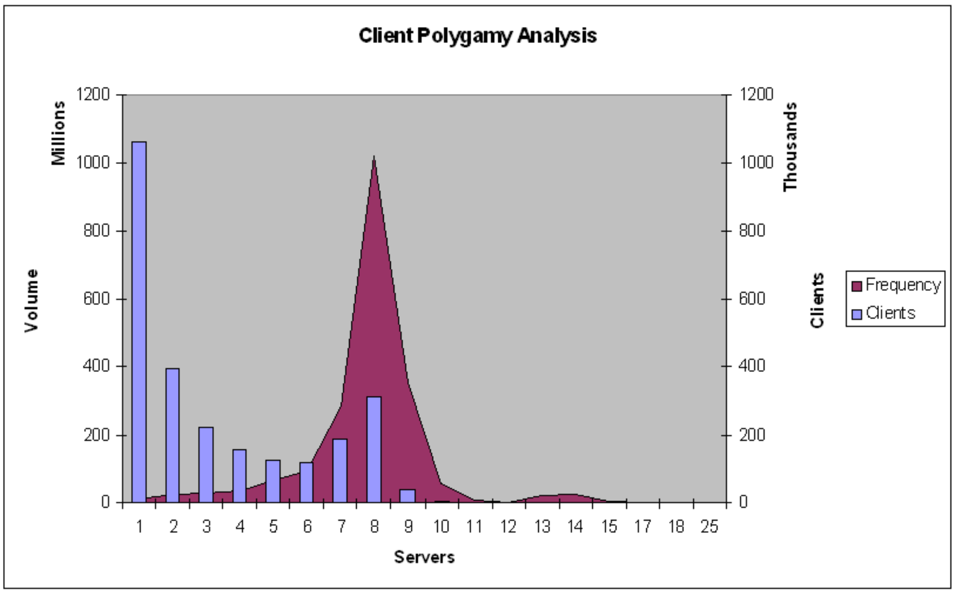


Figure 5.1. Diagram showing the polygamy (number of servers used) by clients, the number of clients and amount of traffic represented by those clients.

Figure 5.1 shows the polygamousness of clients as represented in the collected data. The blue bars show the number of clients, and the maroon area displays the

المنارة للاستشارات
www.manaraa.com

volume of traffic represented by those clients, for each of the server counts shown below. For example, there are over a million clients that are monogamous (only utilize one DNS server during the entire 10-day period), however, all of these clients combined represent a tiny fraction (0.61%) of the total volume of requests. Whereas, clearly, the vast majority of traffic (over a billion requests) was observed by nearly 12% of the clients that utilize exactly eight DNS servers.

Client polygamy as it relates to the traffic generated by individual clients and groups of clients must take into consideration the filling of server holes as discussed previously (we will examine how the client data was adjusted shortly). To illustrate how significant the affect the client adjustment process had on the dataset, we isolated the client in the dataset with the most traffic. This client was bound to 14 different DNS servers (for which we have data). Keep in mind, it could have been bound to other servers for which we do not have data. Figure 5.2 shows the distribution of this client's traffic across the 14 servers.

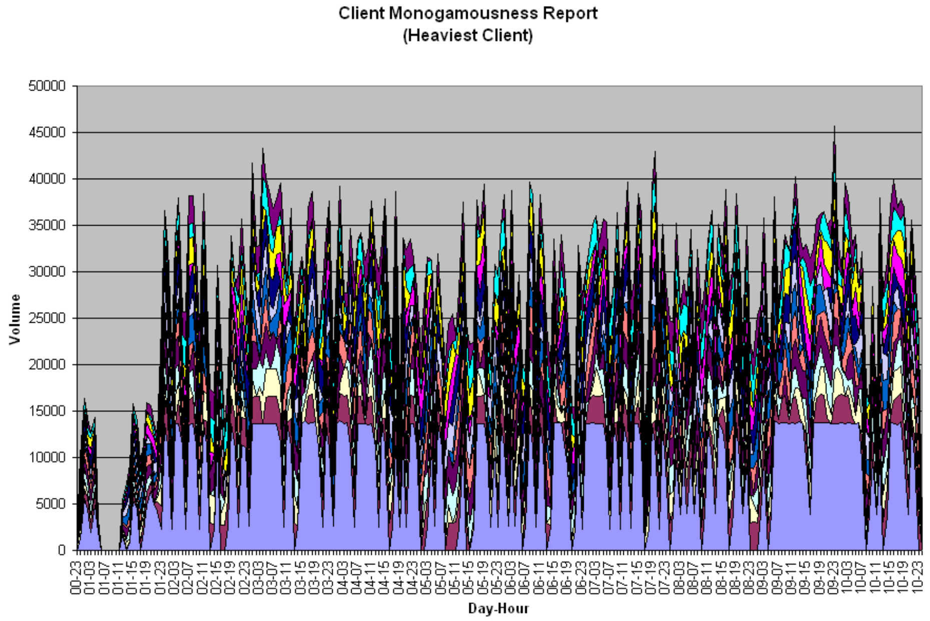


Figure 5.2. Traffic distribution of one client across all its servers for the 10-day period.

After the data was adjusted based on the algorithm that will be presented in

المنارة للاستشارات
www.manaraa.com

section 5.2, the data becomes much smoother and presents a much more consistent behavior pattern, as can be seen in figure 5.3.

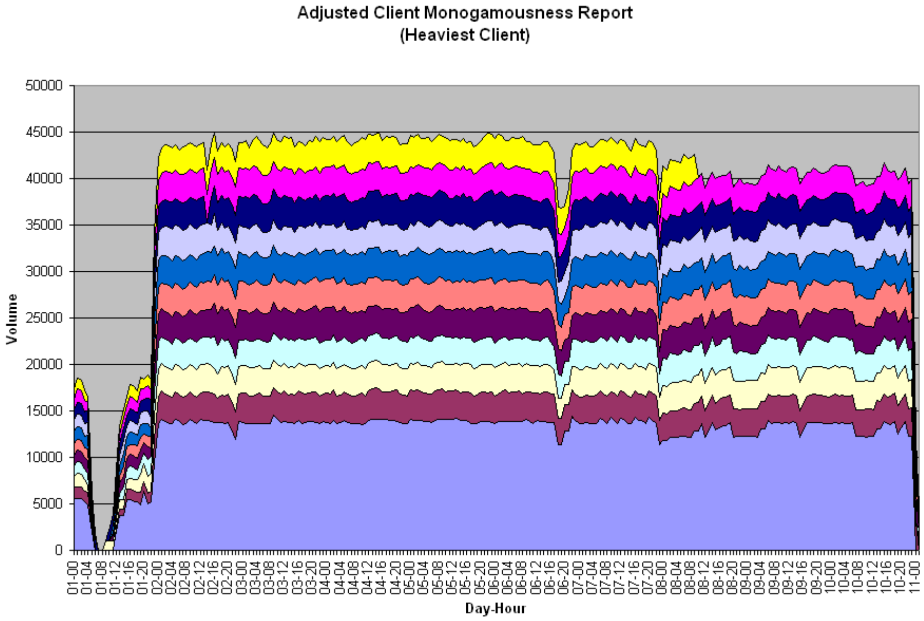


Figure 5.3. Adjusted traffic distribution of one client across all its servers for the 10-day period.

5.2. CLIENT TRAFFIC ANALYSIS

With the server traffic volume data in hand, we began to analyze the client traffic. The method used to adjust the client data parallels the adjustments made to the server traffic and uses those results. The primary difference between adjusting the server traffic and adjusting the client traffic is that there are many more clients (over 2.5 million) to process than there are servers (26), and since each client must be processed for each server to which it is bound (this is where client polygamy becomes an issue), this increases the list to over 8.5 million (remember from figure 5.1 that the bulk of clients talk to as many as 8 different servers for which we have data).

The traffic for different clients can exhibit very different patterns. For the client traffic analysis, it is helpful to aggregate numerous clients together in order to start to

www.manaraa.com

see similar patterns form. The trick is to try to be clever so that clients with a similar pattern are grouped together. Intuition tells us to start by grouping high-volume clients together, and low-volume clients together. Later, we can verify our intuition by analyzing the variance of certain clients from the aggregate pattern developed. The approach we took for grouping clients involves examining the traffic volumes calculated for each client and assigning a percentile rank. From there, several options are possible for aggregating the clients. First, we could sort the clients based on the total volume each client generates and then divide them into 10 even groups where each group contains the same number of clients. The problem with this approach is that the highest-volume clients will be disproportionally represented over the lowert volume clients. The grouping we used involves dividing the groups into deciles (groups of 10 percentiles) based on cumulative traffic volume (see table 5.1). Each decile group represents roughly 10% of the total volume of requests. Even though the number of clients in each group varies, each group is equally represented based on volume of traffic.

Decile	Clients Per Decile	Min Volume	Max Volume	Aggregate Volume
1	2477307	0	1379	202886210
2	80322	1380	4708	202806972
3	28813	4709	10618	202825584
4	13958	10619	20051	202841854
5	7756	20052	34542	202830513
6	4639	34543	55950	202843126
7	2930	55951	87116	202820052
8	1860	87117	138966	202893117
9	1177	138967	215926	202803433
10	440	215927	6901682	202833014

Table 5.1. Client decile groups each representing (roughly) equivalent cumulative traffic volume.

Once the client decile groups have been established, we aggregate the traffic for each client decile group by server. At this point, each server must be checked for a

المنارة للاستشارات
www.manaraa.com

hole. If a hole exists, then the volume for that hour must be prorated to account for the missing traffic (from the discussion previously on data sufficiency, we are only working with data from servers with holes due to missing log data). The entire algorithm is outlined below.

Step 1: Aggregate volume for the clients in the client decile group by server and for each server, look up minutes of data and scaling factor values.

Step 2: Process servers individually. First, sort days based on server minutes of data available.

Step 3: Examine each decile-server-day-hour value and compute adjusted volume. As before, the first day is handled differently than the others. Where a scaling factor is greater than zero, we multiply the scaling factor by the volume to get the adjusted volume. If hours are missing at the beginning and the end of the first day, we simply copy values from the first available or last available hour, respectively. If hours are missing in the middle of the data for the first day, we simply interpolate the missing values from the prior and next available hours.

For subsequent days, we revisit the algorithm and results from server traffic analysis. There are two cases to consider when examining a given client-server-hour volume value:

1. The server-day-hour scaling factor is non-zero. This means that the traffic value we have for this decile-server-day-hour must be multiplied by the scaling factor to get an adjusted decile-server-day-hour value. The volume could be zero if the clients in this decile group generated no traffic to that server during that hour, in which case the adjusted value will still be zero.

2. The server-day-hour scaling factor is zero. This indicates that either we have no data from the server for this hour or the number of minutes of data for that server-hour is less than our established threshold of 50%, in which case we considered this an unreliable hour and treated it as if it was a missing hour. This means that we interpolated the value for the server traffic using formula (4) and so we must likewise interpolate the value for the decile-server-day-hour. For a particular decile-server-day-hour value, the formula to compute the adjusted value is to add up the decile-server-day-hour values for this day where the server

المنارة للاستشارات
www.manaraa.com

scaling factor is not zero, multiply that times the sum of the adjusted server volumes for this hour, then divide the result by the sum of the adjusted server volumes for hours with non-zero scaling factors this day. As with the server adjustment process, this gives an updated set of decile-server-day-hour traffic volume values that takes the volume of traffic of the day as well as the relative traffic volume of this hour of the day from other days into consideration. In order to ensure reliability, we once again processed each day in order from most reliable to least reliable based on total minutes of log data available for each day.

For each of the server-day-hour values processed above we add up the adjusted volume of traffic, and this gives the aggregate adjusted volume of traffic for the client decile group. From the adjusted client traffic values, we can begin to examine the traffic pattern for a "normal" heavy client. As with the server analysis, there is a dramatic improvement in regularity between the adjusted and unadjusted datasets. Figure 5.4 shows the aggregate traffic for one client decile group (consisting of the clients that generate the top 10% of the total volume of traffic) across all of the servers to which it is bound (for which we have data).

As before, we next examine figure 5.5 which shows the traffic for the same group of clients once it has been adjusted to account for the holes in the server data using the technique described above.

Next, we examine what this analysis shows for other client decile groups, such as the decile group representing the clients with the lightest traffic volumes.

Figure 5.6 shows the aggregate traffic for one client decile group (consisting of the clients that generate the top 10% of the total volume of traffic) across all of the servers to which it is bound (for which we have data).

As before, we next examine figure 5.7 which shows the traffic for the same group of clients once it has been adjusted to account for the holes in the server data using the technique described above.

5.3. DOMAIN TRAFFIC ANALYSIS

Next, we analyze the traffic for each domain name requested. The method used to aggregate and adjust the domain name data is nearly identical to the adjustments

المنارة للاستشارات
www.manaraa.com

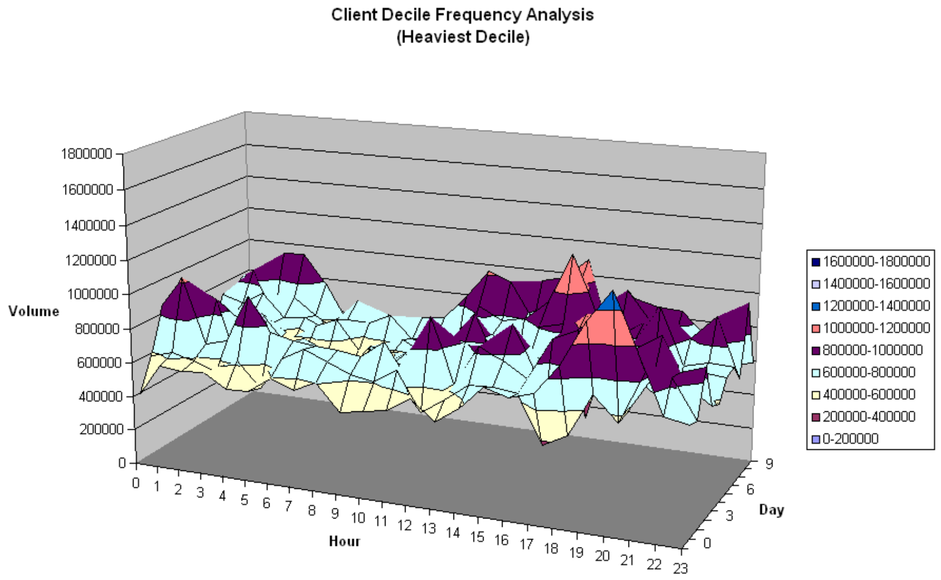


Figure 5.4. Traffic for heaviest client decile group for the 10-day period.

made to the client traffic and again uses the results of the server analysis. The only difference between adjusting the domain name traffic and adjusting the client traffic is using the domain name column instead of the client IP address column in the database. As with the client processing, there are many domain names (over 3 million) to process.

For the domain name traffic analysis, it is once again helpful to aggregate numerous domain names together in order to examin the patterns between like domain names. As with the client processing, we group high-volume domain names together and low-volume domain names together. The grouping we used involves dividing the domain names into deciles (groups of 10 percentiles) based on cumulative traffic volume (see table 5.2). Each decile group represents roughly 10% of the total volume of requests. As with the client decile groups, the number of domain names in each group varies, but each group is equally represented based on volume of traffic. A slight difference with the domain names, however, is that there is more of a discrepancy between the volume requested of high frequency domain names and low frequency domain names. In the table, the top three deciles are dominated each

www.manaraa.com

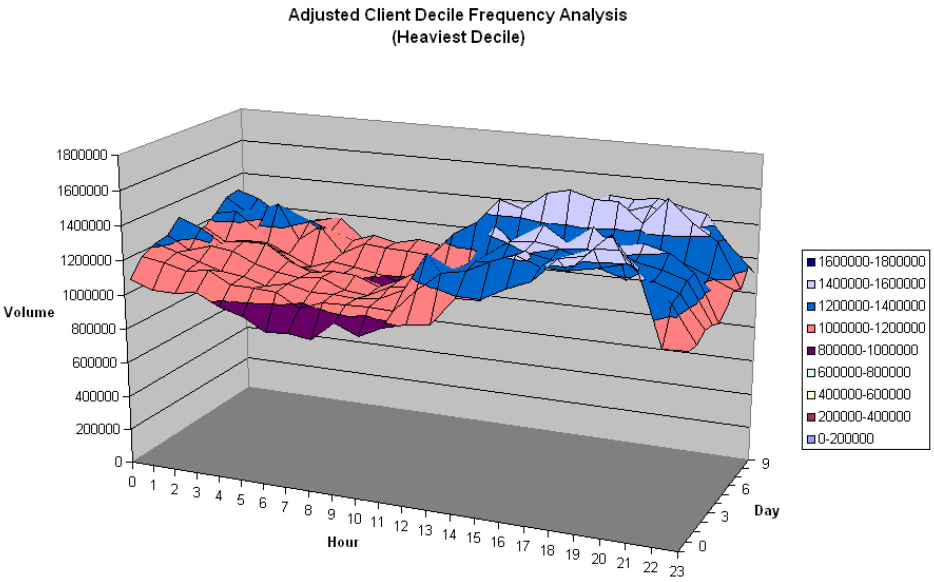


Figure 5.5. Adjusted traffic for heaviest client decile group for the 10-day period.

by only one domain name, and the fourth decile aggregates only two domain names. This leaves a disproportionate number of domain names in the tenth decile. However, for our purposes, this doesn't constitute a significant problem and we could always reproportion the domain names into alternate groups manually.

Once the domain name decile groups have been established, we aggregate the traffic for each domain name decile group by server. At this point, each server must be checked for a hole. If a hole exists, then the volume for that hour must be prorated to account for the missing traffic (from the discussion previously on data sufficiency, we are only working with data from servers with holes due to missing log data). The entire algorithm is omitted, due to the fact that it is exactly the same as the algorithm used in client processing.

At the end of processing each of the server-day-hour values above we can begin to examine the traffic pattern for a "normal" heavy domain name. As with the client and server analyses, there is a dramatic improvement in regularity between the adjusted and unadjusted datasets. Figure 5.8 shows the aggregate traffic for one domain name decile group (consisting of the domain name in the first decile that

المنارة للاستشارات
www.manaraa.com

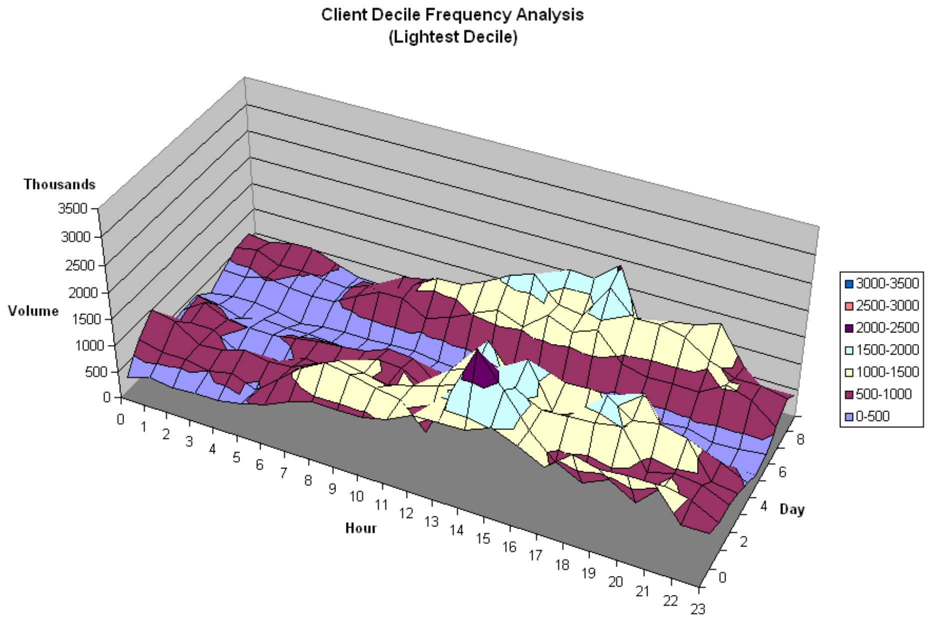


Figure 5.6. Traffic for lightest client decile group for the 10-day period.

generates the top 10% of the total volume of traffic) across all of the servers from which the domain name was requested (for which we have data).

As before, we next examine figure 5.9 which shows the traffic for the same domain name once it has been adjusted to account for the holes in the server data using the technique described above.

Next, we examine what this analysis shows for other domain name decile groups, such as the decile group representing the domain names with the lightest traffic volumes.

Figure 5.10 shows the aggregate traffic for one domain name decile group (consisting of the domain names that generate the top 10% of the total volume of traffic) across all of the servers from which the domain name was requested (for which we have data).

As before, we next examine figure 5.11 which shows the traffic for the same group of clients once it has been adjusted to account for the holes in the server data using the technique described above.

المنارة للاستشارات
www.manaraa.com

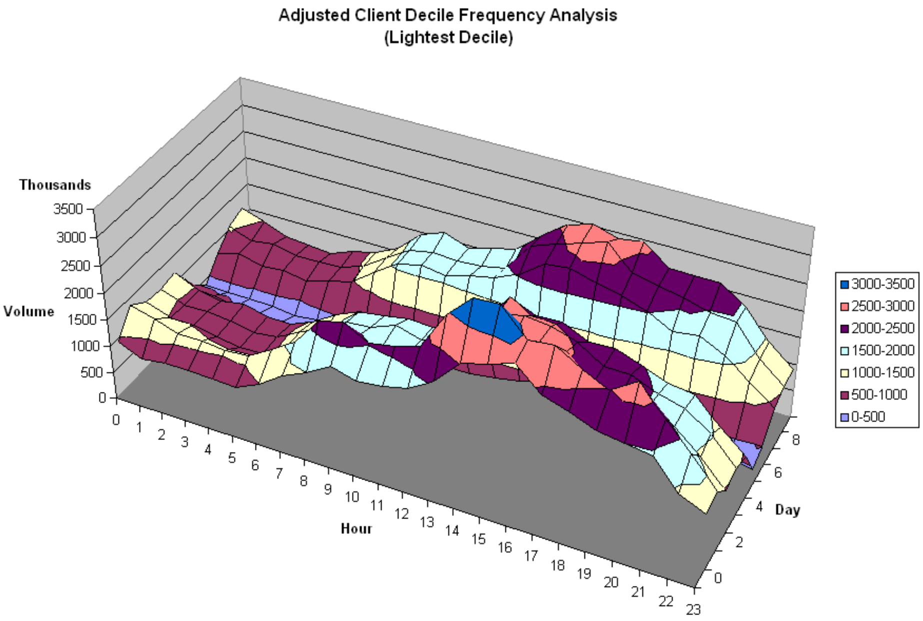


Figure 5.7. Adjusted traffic for lightest client decile group for the 10-day period.

Decile	Domains Per Decile	Min Volume	Max Volume	Aggregate Volume
1	1	265940087	265940087	265940087
2	1	142193910	142193910	142193910
3	1	121219666	121219666	121219666
4	2	94370237	116432237	210802474
5	4	42150988	91905649	252276144
6	7	27443752	37836078	217956083
7	9	18378494	27057672	203110531
8	20	6499397	15942293	208851647
9	50	2095633	6496745	201621665
10	3044001	1	2062561	204411668

Table 5.2. Domain name decile groups each representing (roughly) equivalent cumulative traffic volume.

www.manaraa.com

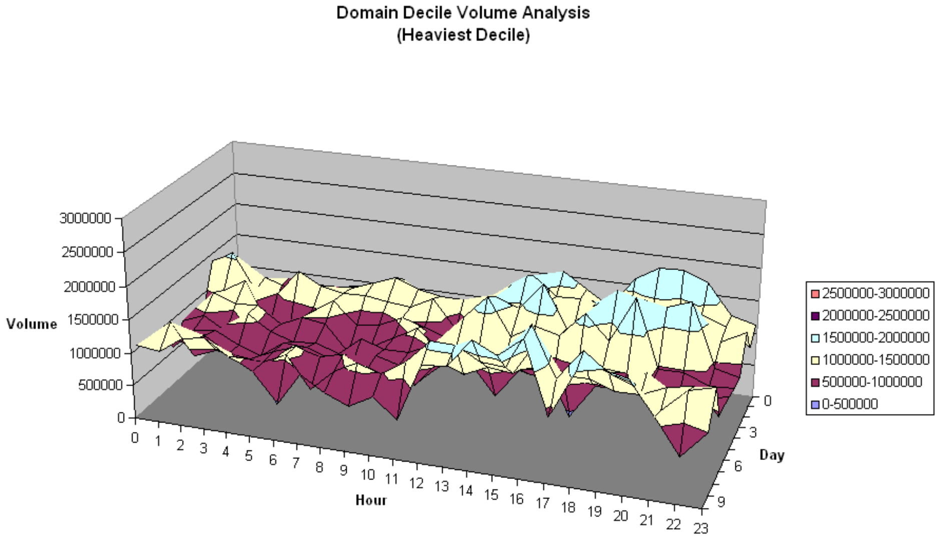


Figure 5.8. Traffic for heaviest domain name decile group for the 10-day period.

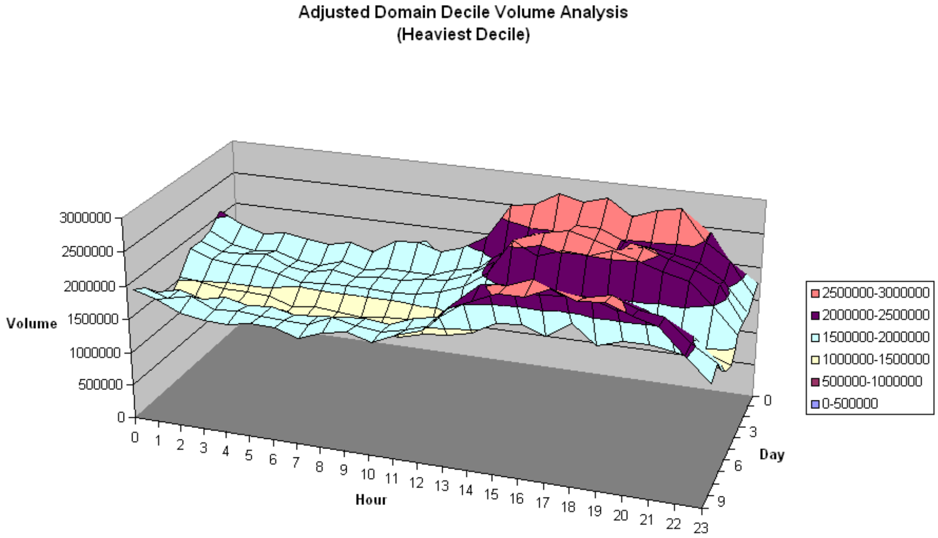


Figure 5.9. Adjusted traffic for heaviest domain name decile group for the 10-day period.

www.manaraa.com

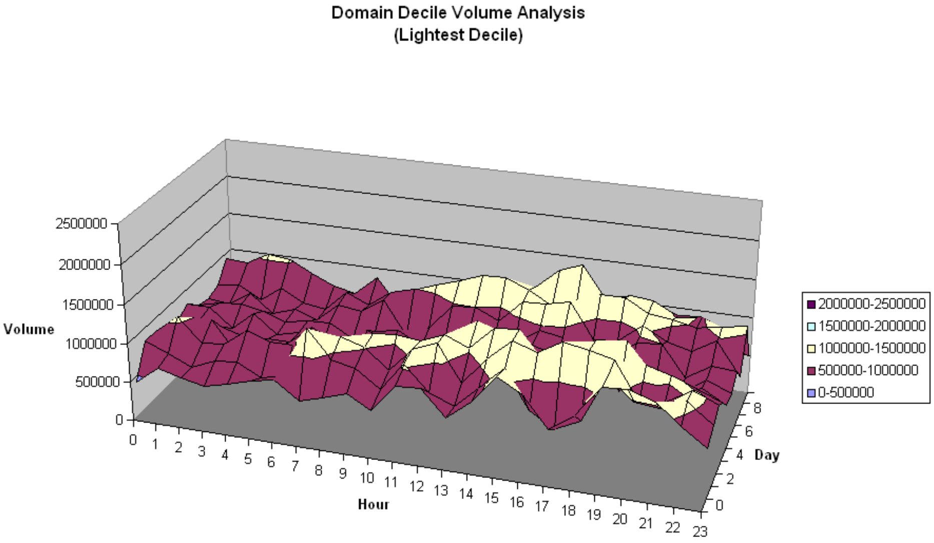


Figure 5.10. Traffic for lightest domain name decile group for the 10-day period.

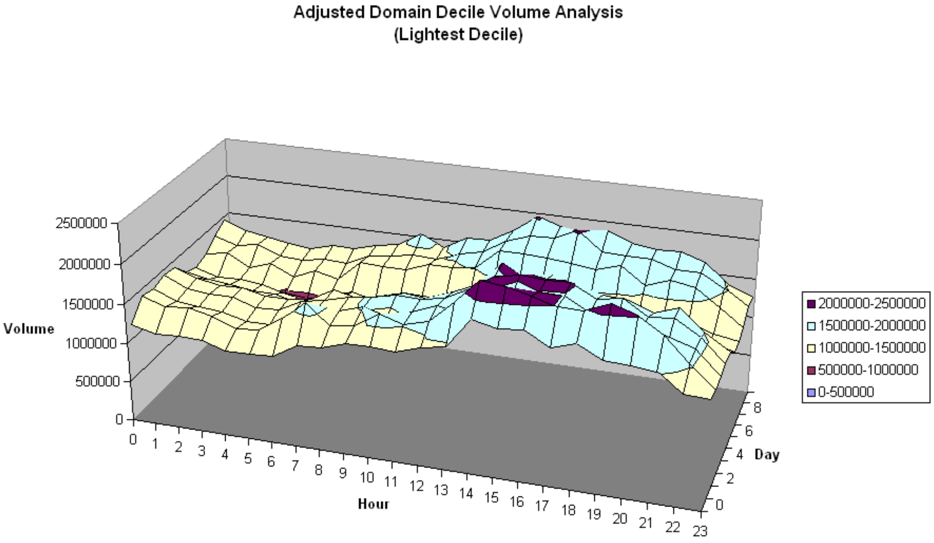


Figure 5.11. Adjusted traffic for lightest domain name decile group for the 10-day period.

www.manaraa.com

6. CONCLUSION

We started with a set of raw DNS logs covering a period of time and capturing volumetric data regarding frequency of requests of a set of DNS servers. However, this data was not useful in its raw form because of gaps in coverage. After preprocessing the data and cleaning it using the demonstrated techniques, we now have a robust set of data that can be analyzed in a flexible manner because the data is stored in a relational database. The data has been cleaned of unwanted request types and bad data in the DNS logs that were skewing the statistics. The data has been repaired to account for holes in the source data. Finally, new data has been produced that will assist in future analysis centered around detecting anomalous DNS server traffic. DNS servers provide a critical function in directing Internet traffic. By detailing a reproducible method for analyzing server log data to identify habitual patterns for the traffic processed by DNS servers, we have laid the groundwork for tackling the problem of detecting attacks on Domain Name Service (DNS) servers, which centers around properly identifying illegitimate traffic.

المنارة للاستشارات
www.manaraa.com

BIBLIOGRAPHY

[1] Everhart, C., Mamakos, L., Ullman, R., and Mockapetris, P. New DNS RR definitions; RFC-1183. *Internet Request for Comments*, 1183 (Oct. 1990).

[2] Gustafsson, A. Handling of unknown DNS resource record (RR) types; RFC-3597, Sept. 08 2003.

[3] Han, J., and Kamber, M. *Data Mining: Concepts and Techniques (The Morgan Kaufmann Series in Data Management Systems).* Morgan Kaufmann, September 2000.

[4] M. E. Snyder, R. S., and Thakur, M. A game-theoretic framework for bandwidth attacks and statistical defenses. In *LCN '07: Proceedings of the 32nd Annual IEEE Conference on Local Computer Networks* (Washington, DC, USA, 2007), IEEE Computer Society.

[5] Naraine, R. Massive ddos attack hit dns root servers. *www.internetnews.com/dev-news/article.php/1486981* (Oct. 2002).

المنارة للاستشارات

www.manaraa.com

VITA

Mark Edward Snyder was born May 5, 1965 in Independence, Missouri, USA. Mark earned a secondary degree from Fort Osage High School, as well as a certificate for 1080 hours of computer training from Fort Osage Vocational-Technical School, both in May, 1983. From there he went on to earn a Bachelor of Science degree in Computer Science from Central Missouri State University, Warrensburg, Missouri, in May, 1987. After a number of years as a professional software developer for such companies as American Airlines and Microsoft, Mark became a cattle farmer, fireman, and instructor of Computer Science at Ozarks Technical Community College. Once again realizing that computer science was his true passion, Mark went back to school to pursue an advanced degree, earning a Master of Science in Computer Science from the Missouri University of Science and Technology in Rolla, Missouri, in December, 2008.

المنارة للاستشارات
www.manaraa.com