CERIAS Tech Report 2016-6

Computational Environment for Modeling and Analysing Network Traffic Behaviour using the Divide and Recombine Framework

by Ashrith Barthur
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ashrith Barthur

Entitled

COMPUTATIONAL ENVIRONMENT FOR MODELING AND ANALYSING
NETWORK TRAFFIC BEHAVIOUR USING THE DIVIDE AND RECOMBINE FRAMEWORK

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

William S. Cleveland
Chair

J. Eric Dietz

Bowei Xi

Eugene H. Spafford

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): William S. Cleveland

Approved by: Eugene H. Spafford 10/4/2016
Head of the Departmental Graduate Program Date

COMPUTATIONAL ENVIRONMENT FOR MODELING AND ANALYSING NETWORK TRAFFIC BEHAVIOUR USING THE DIVIDE AND RECOMBINE FRAMEWORK

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Ashrith Barthur

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2016

Purdue University

West Lafayette, Indiana

“When mountain-climbing is made too easy, the spiritual effect the mountain exercises vanishes into the air.”

— *D.T. Suzuki*, The Training of the Zen Buddhist Monk

Dedicated to life-experience, the greatest *Guru*.

And every *being* that has been a part of this experience.

ACKNOWLEDGMENTS

The work of the scientific research community has been foundational in my research study, therefore my foremost acknowledgement goes to them.

I will always be indebted to three people: Dr. Victor Raskin, Dr. Gita Ramaswamy, and Dr. Sonny Ramaswamy; who very meticulously channeled my curiosity and love for research towards academic research.

My time in West Lafayette would be uneventful if not for my friends and fellows members of the tea club: Arnab, Venkatesh (Nathan), Himanshu, Parijath, Shekar, Saumitra, and Sambit. The extended discussions in the field of physics, politics, religion, tea, etc. have shaped my work and personality. I will always be grateful for making all-nighters exciting, and Indiana winters inspiring.

I am beholden to my friends Preeti and Sarath; who have supported me through tough times with knowledge and advice.

Due regards to my cousins, Sitaram and Rupali, who have been my support structure for the last nine years.

My thesis and, by extension, my research would not be done if not for the support of Saptarshi and Yotam. Their expert proof reading capabilities kept the draft versions to a minimum, and their advice made the thesis wholesome.

Adam, Doug, Keith, and Mike, thank you for putting up with the most unusual requests. Eleventh hour poster printing, CAT6 cable crimping, migrating tera-bytes of data across campus, and simulating a Botnet should definitely classify as unusual.

It would be fair to say that without John Gerth's support I would probably not have access to interesting datasets. I will always be grateful to him for providing me effective advice on code and algorithm design.

I owe much of my success with thesis completion to Marlene Walls. Her assistance with the administrative maze has been immeasurable. Her lasting advice on perfection reminds me of a quote by Margaret Atwood, *"If I waited for perfection... I would never write a word."*

Akhila, Ma, and Pa, thank you for your patience. This is as much yours as it is mine.

I would like to thank my thesis committee for their unyielding support and insightful comments. Dr. Xi, thank you for the challenges, I would have learnt little if not for them. Dr. Dietz, thank you for reminding me that milestones matter as much as the end result. Dr. Spafford, I will always remember that the right question matters, sometimes much more than the answer itself.

Finally, I would like to thank my advisor, Dr. William S. Cleveland. His commitment to guide students is unwavering and unmatched. His dedication to research is one-of-a-kind. As I look back at my time at Purdue, I realise that I have naturally absorbed his philosophy of data analysis and data visualization. I am eternally grateful to him, for his guidance and advice; and his strive for perfection.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Barthur, Ashrith Ph.D., Purdue University, December 2016. Computational Environment for Modeling and Analysing Network Traffic Behaviour Using the Divide and Recombine Framework. Major Professor: William S. Cleveland.

There are two essential goals of this research. The first goal is to design and construct a computational environment that is used for studying large and complex datasets in the cybersecurity domain. The second goal is to analyse the Spamhaus blacklist query dataset which includes uncovering the properties of blacklisted hosts and understanding the nature of blacklisted hosts over time.

The analytical environment enables deep analysis of very large and complex datasets by exploiting the divide and recombine framework. The capability to analyse data in depth enables one to go beyond just summary statistics in research. This deep analysis is at the highest level of granularity without any compromise on the size of the data.

The environment is also, fully capable of processing the raw data into a data structure suited for analysis.

Spamhaus is an organisation that identifies malicious hosts on the Internet. Information about malicious hosts are stored in a distributed database by Spamhaus and served through the DNS protocol query-response. Spamhaus and other malicious-host-blacklisting organisations have replaced smaller malicious host databases curated independently by multiple organisations for their internal needs. Spamhaus services are popular due to their free access, exhaustive information, historical information, simple DNS based implementation, and reliability. The malicious host information obtained from these databases are used in the first step of weeding out potentially harmful hosts on the internet.

During the course of this research work a detailed packet-level analysis was carried out on the Spamhaus blacklist data. It was observed that the query-responses displayed some peculiar behaviours. These anomalies were studied and modeled, and identified to be showing definite patterns. These patterns are empirical proof of a systemic or statistical phenomenon.

CHAPTER 1. INTRODUCTION

Domain Name System or DNS is a vital protocol for the operation of the internet. DNS makes it easy for users to connect to services across the internet using “domain names,” which are made up of names, numbers, words, and phrases which can be easily recalled by users.

DNS is a large, distributed database across the internet. It maps an easily recallable domain name to IP address - which is the actual identifier needed to reach a host and its service. However, IP addresses are difficult to remember due to their complicated structure. This problem is solved by DNS which maps IP addresses to easily recallable domain names.

DNS is also used for purposes other than domain name translation; these include:

1. DNSSEC is a certification service that runs on DNS. It verifies the identity of a host using certificates.

2. DNS is used to blackhole known malicious domain requests, thus containing software infections.

3. DNS is used as a communication protocol to provide information about malicious hosts.

DNS serves the users of the internet in multiple ways, thus making it a vital protocol for the operation of the internet.

In this research, Spamhaus query blacklist information is studied. This blacklist information contains the identities of malicious hosts. The information is provided as part of a query response process; interested hosts query Spamhaus servers requesting records of malicious activities of specific hosts. This service is studied in depth to identify peculiar behaviours. These behaviours are studied and modeled to understand the underlying phenomenon.

Spamhaus is the largest malicious host information provider on the internet. The data for the analysis of this thesis work is collected at the Spamhaus mirror server installed at the Department of Computer Science at Stanford University. Raw DNS data which contain information about malicious hosts, are collected as queries and responses. These query and response transactions occur between querying hosts and the Spamhaus server. The data is prepared by removing corrupt and incomplete records for an accurate analysis. The refined data is reconstructed into a carefully defined data structure that is robust and enables different types of data analyses.

The analytical environment for deep, complex analysis in cybersecurity is constantly evolving. A large part of this evolution can be attributed to past research. The past research includes:

1. SSH based key stroke analysis
2. SSH based inside host detection
3. ICMP ping trace study
4. Root DNS query response study

This analytical environment contains a comprehensive toolset for analysing packets of different network protocols. It uses R as its analytical language, Hadoop as its data store and processing engine, and Tessera (formerly known as RHIPE) as its analytical framework. The environment is named *BoHD - Built on HaDoop* and has been implemented at the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University.

CHAPTER 2. SPAMHAUS

Spamhaus is an organisation that provides malicious host information to a requesting entity - host or user. The information provided is a detailed host malicious activity report, and is structured and stored on a distributed infrastructure. Both the request for information and the information itself are transmitted between the requesting host and Spamhaus servers by encapsulating them in the DNS protocol query-response packet. All this information is served from the largest, distributed DNS server infrastructure in the world (The Spamhaus Project Limited, 2014).

The malicious host information service provided by Spamhaus and similar operators is known as a DNS Block List (DNSBL).

DNSBL services from Spamhaus and other providers have replaced the similar but smaller databases curated independently by multiple organisations for their internal needs. These services are also popular due to their free access, exhaustive information, historical information, simple DNS based implementation, and reliability. The malicious host information obtained from these databases are used in the first step of weeding out potentially harmful hosts on the internet.

The DNSBL service has replaced a previously existing BGP-based network route black holing approach known as Route Block List, or RBL, which proved less effective (Levine, John, 2010). In particular, the RBL mechanism was inexact as it prevented all communication with the autonomous system (AS) of a malicious host, even when the AS had other non-malicious hosts in it (Vixie, Paul, 2000). In DNSBL, by contrast, blacklist information is transferred over the DNS protocol. Using this information requesting hosts block each malicious host individually, without affecting the communication with other non-malicious hosts. The ability to selectively block a malicious host makes DNSBL the preferred method to prevent spam.

2.1 Malicious Host Classification

Hosts in the Spamhaus database are classified as listed and unlisted hosts. Hosts, that are recorded in the Spamhaus database for their malicious activities are classified as listed hosts. While hosts that do not have any record of their malicious activities in the Spamhaus database are classified as unlisted.

2.2 Block Lists

A listed host in the Spamhaus database will be further classified based on its malicious activities.

A host in the Spamhaus database, whether listed or unlisted is identified by its IP address or domain name. Spamhaus uses these two unique identifiers in order to classify hosts.

2.2.1 Block List Family Classification

IP addresses and domain addresses are associated differently with end hosts.

Studying Spamhaus responses requires Spamhaus query-responses and their corresponding host IP or domain addresses. IP addresses are associated with an end host for a certain period of time. While, domain addresses, could be associated with a single end host for a given period of time, or multiple end hosts at the same time (Brisco, Thomas, 1995). Therefore, the data is separated into two different families by the unique identifiers - IP and domain addresses.

This research focuses only on the host family identified by IP addresses.

2.2.2 Sub-classification of Block List Families

The listed hosts in the IP and domain address families are further classified into classes, where each class identifies a specific malicious behaviour. A host could be listed across multiple classes within a family.

2.2.3 Block List Family by IP Address

The block list family by IP address is divided into four different classes. They are:

2.2.3.1 Spamhaus Block List (SBL)

Spamhaus Block List is the primary class of malicious behaviour identified by Spamhaus.

The Spamhaus Block List is a database of IP addresses. These IP addresses are associated with hosts which are considered responsible for spam, and Spamhaus recommends the querying host to not accept any email from these hosts (Spamhaus, 2016e). This information helps email servers weed out unsolicited, potentially harmful email. SBL, identifies large scale commercial spam operations, spam gangs, and spam organisations. It records spam-friendly internet service providers (ISP), and IP blocks and botnets that are used for spamming. This also includes DNS and email services that support spam.

2.2.3.2 Spamhaus Block List Composite Snowshoe Spam (CSS)

Composite Snowshoe Spam identifies IP addresses of commercial spamming operations that use multiple techniques to keep a light spamming footprint and avoid detection by spam sensors.

CSS detection is targeted at spam operators who typically use a smaller block of IP addresses distributed across multiple IP blocks. They send emails using legitimate-looking domain names generated by using domain generated algorithms. They do not operate as a bulk spammer and send out small number of spam mails to avoid any spam sensor triggers.

Spam operators use small sets of hosts and IP address blocks that are served by a small pool of domain names in a round-robin fashion to send spam. They use a common mail template to send out spam. These spam messages have nothing in common with the domain names that are mailing them.

2.2.3.3 Exploit Block List (XBL)

The XBL is similar to SBL. It is a database that enlists the IP addresses of hosts that have been compromised, and are infected with exploits. These exploits enable the compromised hosts to be used for nefarious purposes, such as, open proxies, built-in spamming engines, email address harvesting, and for further propagating the exploits by finding and infecting newer hosts (Spamhaus, 2016a). The XBL gets its data from many different sources, primarily from the Composite Block List (CBL) that is managed by `www.abuseat.org` (Spamhaus, 2016a).

2.2.3.4 Policy Block List (PBL)

PBL identifies end-host IP address ranges which serve as mail transfer agents (MTA) and send out unauthenticated messages.

Sending out unauthenticated email is treated as non-compliance with ISP's acceptable usage policy and hence identified as a policy violation. End hosts that violate the acceptable usage policy are recorded in the PBL (Spamhaus, 2016d). Some ISPs collaborate with Spamhaus and proactively contribute to the PBL by adding end host IP addresses in their own networks which violate terms of acceptable usage. There are also ISP networks on the internet that do not self-police. In these situations, Spamhaus identifies and records the IP addresses of policy violators by observing spam, botnet and unauthenticated mail traffic from these networks (Spamhaus, 2016d).

Differentiating between violations identified by ISPs and Spamhaus is key to studying Spamhaus's response behaviour in detail. Policy violations are therefore further divided into two subclasses, violations identified by ISPs and those identified by Spamhaus.

2.2.4 Block List Family by DNS Address

The block list family by DNS address is divided into two classes. They are:

2.2.4.1 Domain Block List (DBL) - Spamming Domains

The DBL is a database of domain addresses responsible for spamming. These domain addresses are linked to spamming payloads or could be used as a spamming domains. The domains could also belong to spamming gangs, or commercial spamming operations and be hosting viruses, trojans and worms, etc. that help exploit hosts (Spamhaus, 2016b).

Malicious domains listed in the DBL are used in multiple ways. For example, domains might be used to send out emails or be embedded in the body of a spam message, thus exploiting an end-user by click baiting.

2.2.4.2 Domain Block List (DBL) - Spam Redirection

Spam redirector domains are legitimate but compromised domains that are used to redirect an innocent user or host to a phishing, malware or spamming website (Spamhaus, 2016c).

Spam redirector domains are exploited due to a vulnerability in their web hosting, or internet-facing software. Using this exploit the host is repurposed to redirect every visitor to a malicious website.

2.2.5 Spamhaus Block Listing Period

Malicious hosts are listed on Spamhaus databases for various intervals of time. While the IP address of a spamming host is listed for at least six months, IP address blocks of organised spammers could be listed for several years. Spammers who are detected by CSS are listed for shorter periods - 2 to 3 days, primarily due to their transient behaviour (Spamhaus, 2016f).

The DBL in which malicious domain addresses are listed is recreated every sixty seconds. This is done to accommodate domain churns (Spamhaus, 2016c).

2.2.6 Spamhaus Response Design

The DNSBL uses DNS protocol for transferring information between clients and servers. All the information passed to clients are encoded as DNS messages.

A DNSBL reply with an *A record* (address record) is a confirmation that the queried IP address exists in the database. If the DNSBL replies with a *NXDOMAIN*, or non-existent domain, then the queried IP address does not exist in the database.

The DNSBL uses `127.0.0.0/8` IP addresses to encode block list responses. This IP address block is chosen so that any client that misinterprets these IP addresses as connectable IP address connects only to the local host and does not generate any network traffic (Levine, John, 2010; Krochmal and Cheshire, 2015).

Figure 2.6.5 illustrates a typical query to a Spamhaus server. Here, the client is querying the Spamhaus server for information about a host with the IP address `127.0.0.2` using the `dig` tool. The queried IP address, `127.0.0.2` is reversed and appended to the Spamhaus server's domain address - `sbl.spamhaus.org`. [1] [2] [3]

[1] `127.0.0.2` is not an IP address of a real host on the internet and is used by clients to test connectivity to the Spamhaus server.

[2] The `-t` flag in the dig tool specifics the type of response requested by the client, and here the client requests for an IP address.

[3] The flag `+short` keeps the Spamhaus response short.

```
$ dig -t A 2.0.0.127.sbl.spamhaus.org +short
```

Figure 2.1.: Querying Spamhaus server

Figure 2.6.5 is the Spamhaus response for the query in figure 2.6.5. Here, Spamhaus has identified the queried host as malicious and has replied with `127.0.0.2`, which is the encoded message from DNSBL for any host found in the spam block list.

```
127.0.0.2
```

Figure 2.2.: Spamhaus server response

2.3 Spamhaus Data

The Spamhaus query-response data forms the basis of this research study. It is a collection of client queries and their associated Spamhaus server responses. The queries contain IP addresses of hosts suspected to be malicious. Spamming, distributing malware, and violating ISP policies are activities that are considered malicious by Spamhaus.

The response from Spamhaus informs the querying host if the queried host IP address is listed in the block list. If the response is encoded as an `127.0.0.0/8` IP address then the queried host is listed in the block list and identified as malicious (Krochmal and Cheshire, 2015). If the response is empty with the response code as `NXDOMAIN`, then the IP address is unlisted in the block list.

Every queried host IP address has a corresponding state - listed (malicious) or unlisted (non-malicious/unidentified). By collecting the Spamhaus data for around nine months the changes in the state of a host are studied.

The collected dataset builds relations between the querying and queried hosts across the collection time using the state of the queried hosts. This provides an insight into querying behaviour and the malicious behaviour of hosts.

2.4 Spamhaus Data Collection

The Spamhaus data was collected at the Department of Computer Science at Stanford University. It was collected at the department's gateway by selectively capturing only UDP DNS traffic packets that were sent to and from the Spamhaus mirror situated in the department's network.

The packets were collected by a customized version of *TCPDump* (John Gerth, 2011; TCPDump Working Group, 2014b); captured packets were stored in their native, binary format in *pcap* files (TCPDump Working Group, 2014a). Each pcap file was limited to 1 million packets, thus ensuring easy file handling and distribution. These files were then compressed and moved to an analytical cluster at the Center for Education and Research in Information Assurance and Security (CERIAS) at Purdue University for further processing.

2.4.1 Spamhaus Data Processing Tools

A set of tools streamline the Spamhaus data processing. Three applications, and a custom library together form this set of tools.

The Perl DNS packet processing program is the first that was developed to process Spamhaus data. This program can be used to process Spamhaus and other generic DNS traffic. It is designed to handle the extensibility of a DNS packet using resource records (RR) (Mockapetris, Paul, 1987).

Datasets that are extendible are worrisome due to their variable size. The Perl program handles this concern by identifying and marking logical break points in the data.

For example, resource record (RR) is the data structure used in DNS to provide responses. A sample DNS packet could have N number of Answer resource records[4] and M number of Authoritative resource records[5]. This variability in size creates confusion when designing an analytical data structure for the DNS packet. To accommodate this variability, the Perl code recognises that each section - Query[6], Answer, Authoritative, and Additional[7] always exist in a DNS packet but have a variable number of resource records in them, including times when sections have no resource records. The program recognises and outputs these four sections as separate subsections of data with each section having fixed sized resource records.

The Python `dpkt` packet processing library is adopted for processing Spamhaus DNS packets. This library is customized to support a higher number of decimal spaces in the timestamp; thereby providing an improved level of accuracy for analysis.

The DNS packet processing program is built upon the customized `dpkt` packet library (Barthur, Ashrith, 2014), which decapsulates the raw packet. The decapsulation process separates out the ethernet, IP, UDP and the DNS layers in the packet and extracts specific fields from these layers. For example, the requesting host IP address from the IP layer. Further, the program extracts these fields, converts them from a binary encoding to text, and writes them out to files which are used for reading into the R programming framework.

[4] Answer section in a DNS packet contains the actual DNS answer to the query while other sections would contain the query, the authorized server and additional server information(Mockapetris, Paul, 1987).This could be an IP address, mail server domain, etc. The Spamhaus response is contained in the answer section of the DNS packet.

[5] Authoritative section in a DNS packet contains information about the server that is replying to the query - the server that is authorized to reply to queries.

[6] Query section in a DNS packet contains a copy of the original query sent to the DNS server.

[7] The additional section contains any additional information, IP addresses, domain addresses, etc. that support the data in the answer section.

The DNS layer decapsulation breaks down the entire DNS packet and exposes the entire DNS layer - DNS header, question, answer, authoritative, and additional sections. After multiple iterations of the data structure specific fields are identified as being useful for analysis and are extracted from different DNS sections.

The Python decapsulation extracts specific fields from the DNS packet, writes them as text, and saves them as text output files.

The R program processes the text files in a distributed way. The program runs on the deep analysis and complex data platform Tessera, formerly known as RHIPE [8] (Guha et al., 2012a).

Chunks of the text files are read into the Tessera environment and processed into R objects which are then stored on the HDFS [9].

New fields that identify the blacklist state and the type of blacklist are created in the R objects using the information from the text files.

2.4.2 Spamhaus Data Processing

The Spamhaus data collection process spans nine months and the size of the data is about 1.02 TB. Storing, transferring, processing and storing the processed data has required detailed planning and execution.

Raw binary files were moved from Stanford University to CERIAS, Purdue University as nightly `scp`[10] jobs.

The raw data is stored on BoHD[11], - a twenty node Tessera cluster at CERIAS, Purdue University.

The Python DNS packet processing program runs in parallel on each of the nodes and processes the raw pcap binaries into text files. Each processed text file is

[8] In the Tessera environment data is divided into small size objects. Each object is passed to a task which is part of a larger job. R operations are applied on these data chunks in a distributed manner, and the results are combined in the end. The data is then written back to the HDFS.

[9] Hadoop Distributed File System (Shvachko et al., 2010)

[10] `scp` is an abbreviation for **s**ecure **c**o**p**y, it is a method of copying files across different computers using the `ssh`, secure **sh**ell protocol.

[11] BoHD, is an abbreviation for **B**uilt **o**n **H**a**D**oop

the output of a single pcap binary file, and contains 1 million Spamhaus DNS query-responses.

A bash wrapper script, feeds the pcap binaries to the Python DNS packet processing program and receives the output text files. It then transfers and stores the text files on the HDFS. The files are stored in a directory hierarchy. The raw data is divided into nine folders by month. Each month folder is further divided into folders by the weeks in that month with week folder containing all the text files that belong to that specific week.

The R processing program based on the Tessera *package*[12] reads the text files from each week, processes them in R, and saves them as a *dataframe*[13] in a Hadoop SequenceFile[14]. The SequenceFiles for each week are stored in that specific week's directory.

The raw DNS packet data is collected in time order. Both the Python DNS processing program, and the R processing program retain the time order while processing the raw DNS packet data into text files and R data frame objects. The time order is critical for a time-based DNS query-response analysis.

2.5 Spamhaus Analytical Cluster

The Spamhaus analytical cluster, also known as *BoHD*[15][16], was designed and implemented at CERIAS, Purdue University.

The cluster has 20 enterprise grade nodes, each node has two Intel Xeon CPUs, each CPU has two cores, and each core has two threads. Each node also has 32GB of memory, and 2X 1TB of persistent disk storage space. The entire cluster is

[12] Tessera is both a platform and a package, the package is the library that enables R code to be written in the map-reduce paradigm and run on the Tessera platform.

[13] A data frame is a packagedd set of variables that have similar properties of matrices and lists. It is the most frequently used data structure in R (Hornik, 2016).

[14] SequenceFile is a flat file consisting of binary key/value pairs. It is extensively used in MapReduce as input/output formats (Apache Hadoop, 2009)

[15] BoHD stands for Built on HaDoop.

[16] The cluster was built using a CERIAS resource grant.

connected by a 2 X 1GBps bound link, providing increased network bandwidth under normal operations and redundancy for link failure.

Each one TB disk drive is partitioned three ways. One partition each is dedicated for `/boot`[17], `/`[18], and HDFS. The first two partitions: `/boot` and `/` of size 255MB and 32GB, respectively, are set up with *RAID 1*[19] configuration across the two one TB disks. While the `/boot` is set up on a raw RAID 1 configured partition, the `/` partition is setup on a *LVM - logical volume*[20]that sits on top of the raw RAID 1 configured partition. The remaining raw disk space on each of the 1TB disk is allocated to the HDFS.

Each node on the cluster runs a lean version of Ubuntu 10.04[21]. Each node also has Cloudera Hadoop, the Tessera package and framework that enables deep analysis of large complex data, R programming language and supporting tools and libraries.

The cluster is configured with 1 node as the namenode[22] and jobtracker[23]. A second node is configured as a secondary namenode[24]. 19 nodes are setup as task trackers that are allocated tasks from the jobtracker. They perform R computations utilizing Tessera and MapReduce framework. Out of the 19 nodes 18 of them are also setup as datanodes[25] and contribute to the HDFS where raw, and processed Spamhaus data is stored (Barthur, Ashrith and Crabill, Doug and Tong, Xiaosu, 2015).

[17] The `/boot` partition is used by the bootloader, which loads the operating system.

[18] `"/"` is known as the root partition of a Linux-based machine, it is the start of the directory structure.

[19] RAID - Random Array of Independent Disks, is a data redundancy mechanism using multiple disks. In RAID 1, data on a disk is exactly replicated on one or more disks.

[20] LVM, or logical volume, is a virtual volume that fuses discontinuous raw partitions and presents it as a single continuous volume to the operating system.

[21] Ubuntu is an operating system using the Linux kernel.

[22] Namenode is the main node in Hadoop, it manages ephemeral and persistent data across the cluster.

[23] Jobtracker manages, job chunking and task allocation across worker nodes.

[24] The secondary namenode keeps a copy of namenode's data table. This table contains information about each data block and its location on the HDFS.

[25] Datanodes are managed by the namenode. They store chunks of data as a member of the HDFS. They also provide data to R computations on the same node, or to other nodes in the cluster.

2.6 Spamhaus Dataset Variable Selection

Together, the Spamhaus query and response form a "*transaction*". Independently, but as part of a transaction, they are called sub-transactions.

The variable and data design identifies fields in each sub-transaction that add value to the analysis. It brings these fields together to form a comprehensive data structure for the Spamhaus query-response study. In this design process, multiple fields that add little or no analytical value are eliminated; data collection and storage processes are modified to remove redundancy; data structures are optimized to be insightful and exhaustive.

The Spamhaus data collection process captures the raw packet. This enables access to all the OSI layers in the Spamhaus DNS packet. Fields of value are added to the final dataset from each of these layers.

2.6.1 OSI Data Link Layer, Ethernet Protocol Fields

A Spamhaus query[26] and its respective response pass through multiple nodes that communicate using protocols from layers above the OSI - Data Link Layer, e.g. Internet Protocol. Therefore the query and response are constantly repacked in new ethernet frames. The ethernet frame header contains source and destination MAC addresses, VLAN tags, and a type identifier as seen in figure 2.3. These fields provide no information about the query, querying host, or the response and are therefore omitted from the analysis data structure.

2.6.2 OSI Network Layer, IP Protocol Fields

Encapsulated in the ethernet frame is an IP packet, figure 2.4. The IP header has many fields, but only two fields - source address and destination address, provide the IP addresses of the querying host and the Spamhaus server.

[26] A Spamhaus query is sent from the querying client to the Spamhaus server.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source MAC Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                     Destination MAC Address                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                802.1Q Tag                         | Eth Type  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.3.: Ethernet Frame Header (Postel, Jon, 1984)

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version|  IHL  |Type of Service|          Total Length         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Identification        |Flags|      Fragment Offset    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Time to Live |    Protocol   |         Header Checksum       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Source Address                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Destination Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Options                    |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2.4.: IP Packet Header (Postel, Jon, 1981)

The source and the destination addresses are reversed in the query and response IP packets. Additionally, Spamhaus server has a constant, publicly published IP address. Therefore, to build a comprehensive data structure it is sufficient if the querying IP header from the query or the response is captured.

2.6.3 OSI Transport Layer, UDP Protocol Fields

The UDP datagram header contains four fields - source port, destination port, length (data length), and checksum, as seen in figure 2.5. The two fields that are valuable are the source and the destination ports. The UDP data length provides no information about the query-response as the Spamhaus DNS query-response has a very rigid format. Similarly, the UDP checksum too has no analytical value.

Similar to the IP packet, the source and destination ports are reversed in the query and response UDP datagrams. While, the port number used by the querying host could vary, the Spamhaus server port number is constant and set to 53. Therefore, collecting querying host port number from the query or response datagrams would be efficient.

The query host port number is used to identify unique queries along with the querying host IP address. But as the dataset is focused on analysing query host behaviour and contains the actual queries, it does not need the port number to uniquely identify queries. Therefore the entire UDP header is excluded from the dataset.

2.6.4 OSI Application Layer, DNS Protocol Fields

The DNS Protocol data is the data present in the UDP datagram. It provides a structure for the Spamhaus query-response.

The Spamhaus query-response is standardized and therefore there are many fields in the DNS protocol that provide no information about the query or the querying host. As these fields are not essential they are ignored from the analytical dataset.

```
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|                                   |
|          data octets ...          |
+-----------------------------------+
```

Figure 2.5.: UDP Datagram Header (Postel, Jon, 1980)

The DNS protocol can be divided into two main parts, the DNS header and the DNS resource records (RR[27]).

The *ID* field, which is short for transaction identifier, helps in uniquely identifying a DNS query and its associated response, but as the entire DNS query is captured this identifier field is unnecessary and therefore ignored (Mockapetris, Paul, 1987).

The *QR* - query/response field is a flag that differentiates a response from a query. This flag is set to `1` in response and `0` in query. But as a query and its associated response can be identified by other means, e.g. source and destination IP addresses, this field is not needed (Mockapetris, Paul, 1987).

The *Opcode* - operation code establishes the type of query present in the DNS query packet (Mockapetris, Paul, 1987). This field can have five different operation codes, but for the purposes of Spamhaus DNS blacklisting only the *Query*

[27] DNS resource records are data structures provided for DNS queries and response.

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.6.: DNS Header (Mockapetris, Paul, 1987)

- standard query is used. As the analysis focusses only on a constant it does not provide any additional information and is not included in the dataset.

The following are the five different operation codes present in DNS:

1. Standard Query (QUERY)
2. Inverse Query (IQUERY)
3. Server Status (STATUS)
4. Notification (NOTIFY)
5. Update (UPDATE)

The four fields, *AA* - authoritative answer, *TC* - truncation, *RD* - recursion desired, *RA* recursion available provide no valuable information about the query or the querying host, and therefore are excluded from the dataset.

The *RCODE* - response code field provides valuable information about the query. This field has flags that identify correct and incorrect query formatting; the value in this field differentiates address and domain queries; it also has a flag for identifying server response failures. These response codes help differentiate between accepted and rejected queries and the response code field is therefore included in the dataset.

The *QD*, *AN*, *AT*, *AD*, fields provide the number of resource records available in the question, answer, authoritative, and additional section. There counts add no value to the query behaviour and are therefore ignored from the dataset.

2.6.5 Question, Answer, Authoritative, and Additional Sections

The query section in the DNS protocol exists in both, the query and response. In fact, the query section is copied from the query to the response.

The query section of the Spamhaus query-response has three fields: *QCLASS* - query class, *QTYPE* - query type, and *QNAME* - query name. Spamhaus blacklists handle queries that belong to the *Internet* query class, and this study is limited only to IP address type queries. Therefore both fields - query class and query type - are constant and not included in the dataset.

The query name contains the IP address that is being queried and is included in the dataset. This field is in the Spamhaus query format as described in figures , and .

The answer, authoritative, and additional sections in the Spamhaus query are empty and ignored.

Spamhaus responds to a blacklist query in the answer section of the response. The other sections in the Spamhaus response have fields that are of no analytical value and hence are not considered in the dataset.

The answer section in the Spamhaus response is structured as a resource record. Each resource record contains six fields: *NAME*, *TYPE*, *CLASS*, *RLENGTH* - record length, *RDATA* - record data, and *TTL* - time to live. Of these six fields, the name field is a copy of the query name field from the query section. The type field variable is limited to an *A* record as this study is limited to only IP address queries. The class field, as in the query section, is limited to *Internet*. The record length field contains the length of the record data. The time to live field is the length of time the Spamhaus response is valid for, and is constant. These five fields do not provide any direct response information and are therefore not included in the dataset.

The record data contains the Spamhaus response in the form of an IP address, in accordance to the RFC 1060 format (Reynolds and Postel, 1990). This field is included in the dataset as it provides analytical value in studying query-response.

2.6.6 Time Stamp

Timestamp is not a standard field in the entire query-response packet. It is a variable that is provided by the packet collection framework - TCPDump and Pcap library, and hence is available from the pcap binary files. This variable is important to study query-response, querying, and queried host behaviour and is therefore included in the dataset.

2.7 Packet Data Collection and Dataset Optimization

The comprehensive analysis of the Spamhaus DNS packet and its fields highlight that all the fields present in the Spamhaus DNS query are either derivable, or directly available, from the DNS response. Using this information the collection process ignores the entire sub-transaction - Spamhaus DNS query packet, and only collects the Spamhaus DNS response packet.

There are a total of four fields, source IP address from the IP packet, response code from the DNS header, query name from the query section, and record data[28] from the answer section that are included in the dataset. Additionally, timestamp provided by the packet collection process is also included in the dataset. All these variables are extracted by the Python packet processing program and passed to the R data processing program via text files. Four out of the five fields: source IP address, response code, query name and timestamp, occur exactly once per Spamhaus DNS response. The record data field might have multiple occurrences in a single response. The text files accommodate this variability.

2.8 Data Structure Design

The data structure designed for analysis has undergone multiple rounds of redesign and optimization. These changes were made for computational efficiency and to eliminate fields that were analytically not important. Across these different design iterations time has been a very important variable for the analysis. Ordering query-responses by time provides a "*sequence-of-events*" view which is necessary to observe a behavioural progression. And therefore the query-responses have always been maintained in the time order.

The first iteration of the data structure design had a fixed number of query-responses in a subset[29] with time as the subset key.

Each subset is a dataframe with five columns - time stamp, source IP address, response code, Spamhaus query, and Spamhaus response - answer(s). When Spamhaus responds with multiple answers in a response, each answer is stored in a separate, successive row in the subset - dataframe. These successive rows have the same timestamp, source IP address, response code, and Spamhaus query, but different answers.

Spamhaus answers are specific to the class of malicious behaviour. This also includes hosts identified to be non-malicious and whitelisted. The answers identify:

[28] The record data is not a single field and could be multiple records.

[29] In the Tessera framework an subset is a R object which is identified by a unique key (MapReduce Key) and contains a certain number of records, in this case query-response.

1. Spammers
2. Exploited hosts
3. ISP and Spamhaus policy violators
4. Whitelisted email servers

This design was computationally infeasible as the data structure lacked an efficient way of providing summaries by host. Answers belonging to a single query could not be clearly and accurately identified. For every analysis the response code had to be translated.

A second iteration of the data structure design captured all answers of a response in the same row. Host-based summaries were now possible.

Response code had be decoded into three different states: query format error, no error, and non-existent domain.

In the second iteration, each response, along with all the answers, were stored in a single row of the dataframe. This provided the much needed summaries of, and insights into, query-responses. The RCODE was now pre-translated into categories to easily identify query-responses. The Spamhaus answers in the response were grouped into IP spam, IP exploit, IP policy violation, domain spam, IP email whitelisting, and domain email whitelisting. A binary flag (1/0) represented a specific state or value in these groups. For example, 0 represented blacklisted due to generic spam, while 1 represented blacklisted due to Composite Snow Shoe (CSS) spam. Although, this design was a definite improvement than its predecessor, it lacked the ability to differentiate between no information sent, no information available and other categories in the response groups. Therefore a third iteration of the data design was done.

The third iteration expanded on all variables. Each field captured accurate information in the dataset (Barthur, Ashrith and Cleveland, William S. and Gerth, John, 2014). Below is the list of fields that were designed in the third iteration of the data structure design:

1. Time stamp
2. Source IP address
3. IP address or domain queried
4. Query type (IP address, domain, or query error)
5. Response code
6. State of query-response (blacklisted, whitelisted, or no information)
7. IP address spam - indicates spamming hosts, generic spam detection.
8. IP address composite snow shoe (CSS) spam - spammers, using CSS detection.
9. IP address exploit - exploited hosts.
10. IP address ISP policy violation - ISP policy violators.
11. IP address Spamhaus policy violation - Spamhaus policy violators.
12. IP address individual mail whitelist - IP address whitelisted for sending individual mails.
13. IP address transaction mail whitelist - whitelisted for sending transaction mails.
14. Domain address spam - indicates spamming domains.
15. Domain address spam redirection - indicates redirectors to spamming and phishing sites.
16. Domain address individual mail whitelist - domain whitelisted for sending individual mails.
17. Domain address transaction mail whitelist - domain whitelisted for sending transaction mails.

The third iteration of the data structure retains all the data from a response in a single row, making it easy for computation.

Query type identifies IP address, domain, and ill formed queries. State of query shows if the response contained blacklist, whitelist, or no information. Each malicious behavioural class is independently identified with a binary flag. Although this research study is limited to Spamhaus blacklist queries of the IP address type, domain type queries are also stored for future study.

In total, there are 17 fields in each row that uniquely and exhaustively record a Spamhaus blacklist query and response. Each row is part of the subset. A previous research study (Li, Jianfu and Barthur, Ashrith and Cleveland, William S., 2012; White, 2009), suggests that that subset size must be limited to 25% of the Hadoop block size, therefore keeping the number of rows per subset to 6,000.

Optimization of the data structure needs to be verified periodically. This is necessary to identify any loss of information or field redundancy.

Data summaries of the third data structure were observed after six weeks worth of data was collected and processed. The summaries are shown in the following table 2.1:

The number of whitelisted queries was the most noticeable summary in the table. Out of 2.05 billion queries there were 15 queries requesting host whitelist information. This number was the sum total of four different whitelisting classes - IP address individual mail whitelist, IP address transaction mail whitelist, domain address individual mail whitelist, and domain address transaction mail whitelist. The whitelist fields were removed from the data structure as they provided no analytical value. The new data structure had 13 fields after this optimization (**?**). The final data structure is as follows:

1. Time stamp

2. Source IP address

3. IP address or domain queried

Category	Statistic
Collection duration	6 weeks
All queries	2,051,268,078
IP address queries	1,679,915,476
IP address queries with blacklisted result	331,193,986
Domain name queries	370,958,042
Domain name queries with blacklist result	13,093,693
Queries with no available response	1,657,054,290
Whitelisted queries	15

Table 2.1: Spamhaus Collection Summary Statistics (Barthur, Ashrith and Cleveland, William S. and Gerth, John, 2014)

4. Query type (IP address, domain, or query error)
5. Response code
6. State of query-response (blacklisted, whitelisted,or no information)
7. IP address spam - indicates spamming hosts, generic spam detection.
8. IP address composite snow shoe (CSS) spam - spammers, using CSS detection.
9. IP address exploit - exploited hosts.
10. IP address ISP policy violation - ISP policy violators.
11. IP address Spamhaus policy violation - Spamhaus policy violators.
12. Domain address spam - indicates spamming domains.
13. Domain address spam redirection - indicates redirectors to spamming and phishing sites.

CHAPTER 3. DIVIDE & RECOMBINE FRAMEWORK

Divide and Recombine is a new statistical approach designed for the analysis of large complex data. Here the data is divided into subsets and analytic methods are applied to each subset. Each subset is the smallest data unit upon which analytic methods are applied. The output of each analytic method on the subset are combined together to form the final result.

Each analytic method is an independent computation. The computations run in parallel without the need for interprocess communication, which makes them embarrassingly parallel[1].

Using the D&R approach, an analyst applies statistical and visualization methods as computations on subsets of data. Thereby running deep analysis on large complex data.

The D&R approach is efficient and scalable. Virtually any size data can be divided into subsets, analytic methods applied to them, and their results recombined to form the final result (Guha et al., 2012b). The efficiency and scalability of the D&R approach can be optimized by tuning certain computational factors.

Applying an analytic method - statistical, visualization, on a large dataset is time consuming and impractical. The D&R approach solves this problem by breaking down the process of applying a method and obtaining the result into three separate computations. The first computation is an analytical means by which the data is divided into subsets. The second computation takes advantage of the subsets and applies analytic methods - statistical, and visualization, on these subsets in parallel. The third computation applies another set of analytic methods on the results of the subset computation and aggregates the results.

[1] Embarrassing parallel computations are independent in execution and do not need any interprocess communication. Except may be for setting up the analysis and aggregating results (Messina et al.,

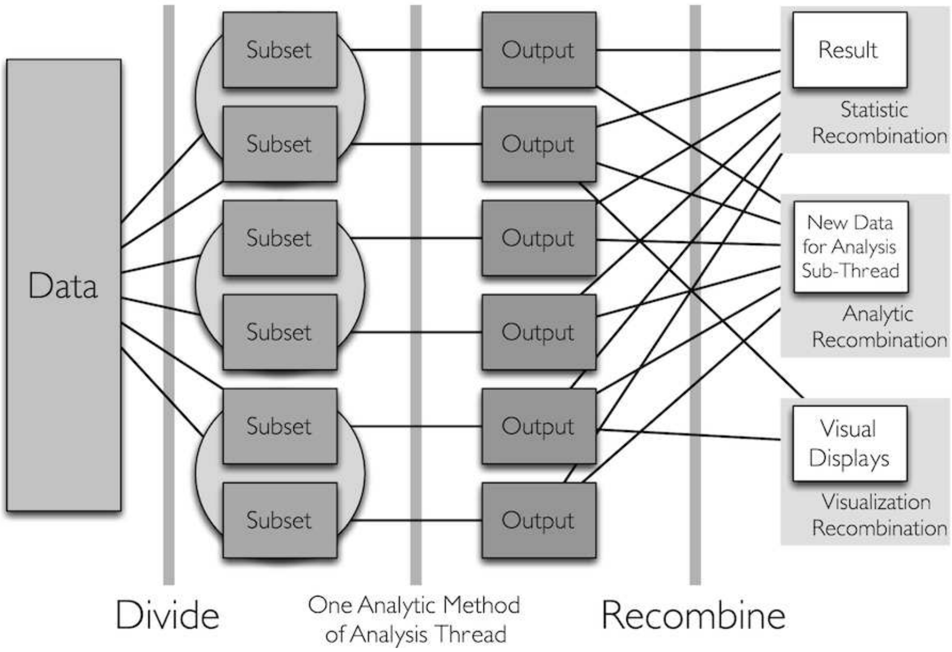

Figure 3.1.: D&R Framework (TesseraIO Research Group, 2015)

The D&R approach is illustrated in the figure 3.1.

The D&R approach is implemented in the D&R framework using R[2] Tessera, and Hadoop. The data is divided into subsets and stored on the HDFS[3], this is the division of data into subsets - R objects in the D&R framework. The Hadoop Map process is the second computation that primarily applies R programming methods, and other program language methods through APIs for R language on the data subsets. The Hadoop Reduce process also applies R and other API-based methods on the output of the Map computation. The Reduce computation further aggregates the results of the applied methods and saves them back to the HDFS.

The Spamhaus DNS blacklisting data is analysed using the D&R approach and framework. The data is divided into subsets in multiple ways, each assisting a unique analytical thread. Each of these subset division encompasses a

1994)

[2] R is a statistical and a visualization programming language (R Core Team, 2016).

[3] HDFS - Hadoop Distributed File System, which is a virtual file system on which Hadoop stores its

comprehensive R data structure. These data structures are inherently designed to work with R methods and functions. Functions outside the R programming language are called using the R programming APIs or native system calls.

3.1 Division Procedures

D&R procedures can be divided into two parts, division and recombination. Further, each part has multiple procedures within itself. In the following subsections two division procedures are discussed: a division procedure adopted for the Spamhaus DNS blacklist data, and another generic D&R divide procedure.

3.1.1 Conditioning Variable Division by Time

The Conditioning Variable Division divides the data on a key variable based on which the data will be analysed. In the Spamhaus DNS blacklist the foundational approach is to study the query-response by time. The Spamhaus DNS blacklist data structure that follows is divided into fixed size subsets - R data frame objects by the conditioning variable: *time*. Each subset is ordered by time, and uniquely identified by the subset's epoch. This is a between subset-variable BSV[4].

1. Time stamp
2. Source IP address
3. IP address or domain queried
4. Query type (IP address, domain, or query error)
5. Response code
6. State of query-response (blacklisted, whitelisted,or no information)

data.

[4] Between subset-variables are variables that have one value per subset (Guha et al., 2012b).

7. IP address spam - indicates spamming hosts, generic spam detection.
8. IP address composite snow shoe (CSS) spam - spammers, using CSS detection.
9. IP address exploit - exploited hosts.
10. IP address ISP policy violation - ISP policy violators.
11. IP address Spamhaus policy violation - Spamhaus policy violators.
12. Domain address spam - indicates spamming domains.
13. Domain address spam redirection - indicates redirectors to spamming and phishing sites.

Each row in the subset represents a Spamhaus response packet. All the rows in a subset are ordered, and are part of a time period starting with the subset's epoch. The timestamps in each time period along with the other twelve variables form the within subset-variables WSV [5].

The thirteen variables in the first conditional variable division by time can be separated into raw variables obtained from the packet and derived variables. Timestamp, source IP address, and response code are the raw variables from the Spamhaus DNS packet. The remaining variables are derived from the Spamhaus DNS query and response.

The first conditioning variable division subsets 13.17 billion Spamhaus DNS blacklist responses which have been collected over a period of nine months.

3.1.2 Conditioning Variable Division by Queried Host

The new, time ordered Spamhaus DNS query subsets are further divided by the queried hosts. This, second division is effective in studying Spamhaus DNS query-responses by time and by queried hosts.

[5] Within subset-variables are variables that vary within a subset.(Guha et al., 2012b)

The analytical thread in this research work only looks at queried host IP address and discounts the domain address queries. Fields that are not relevant to the new dataset - domain address spam and domain address spam redirection - are removed at this step.

The subsets are reorganised by time and queried hosts. The between subset-variables in the new subset structure are the unique queried host IP address and subset's epoch. The rest of the variables in the dataset, including the timestamps form the within subset-variables.

3.1.3 Conditioning Variable Division by Blacklisted Queried Host

The Spamhaus DNS response system replies to multiple queries at any given time. A large part of this response includes not-blacklist responses for queried hosts. A Spamhaus DNS response that is consistently not-blacklist provides no information towards the blacklist behavioural analysis. Therefore, any query host that has not been blacklisted in any of the five blacklist categories at least once in the data is eliminated from the analysis. There are 207 million unique queried hosts before the elimination process. Post elimination, there are 59 million unique queried hosts. These hosts form the data in the conditioning variable division by blacklisted queried host.

In this final data division the between subset-variables are the subset's epoch and queried hosts that have been blacklisted. The rest of the variables from the second conditioning variable division are retained as the within subset variables. This dataset forms the basis of the Spamhaus DNS blacklist analysis.

3.1.4 Replicate Division

The replicate division is adopted when multiple observations of a distinct number of variables in the data need to be partitioned. Random-replicate (RR) is a sub-procedure of the replicate division that uses a random sampling approach to partition the data.

Near-exact-replicate (NER) is another sub-procedure of the replicate division which breaks the observations into local neighbourhoods, with each neighbourhood having an almost-equal number of observations in it. A representative point is chosen from each of these neighbourhoods (Guha et al., 2012b).

3.2 Recombination Procedures

There are three different recombination procedures. Each has been used for analysing and understanding the Spamhaus DNS blacklisting.

3.2.1 Statistic Recombination

The number of queries for each queried host is a good statistic that assists in understanding the data. Therefore, a statistic recombination sub-procedure is applied on the data to extract this information for further study.

The subsets are divided by queried hosts and epochs. The number of queries with blacklist responses are calculated in each subset, and then they are re-combined by queried hosts. Recombining a statistic across subsets is known as statistic recombination.

The distribution of this statistic is studied in detail to understand focus areas of the data. These focus areas would be cognostic in understanding Spamhaus DNS blacklist behaviour.

3.2.2 Visual Recombination

The statistic recombination outputs the number of queries for 59 million hosts - the number of hosts that have been blacklisted at least once. These counts form the distribution of the blacklisted queries by queried hosts. A subset of this data is obtained by sampling the distribution. The sampled data is a representative dataset for the underlying blacklisted queried host population.

The sample data is visualized by a visualization recombination sub-procedure using quantile plot visualization. The empirical quantiles are plotted against quantiles of a mathematical distribution. These distributions are one among uniform, normal, or gamma (Cleveland, 1993). The visualization recombination method provides a mechanism for rigorous visualization, at detail, and at the finest granularity (Guha et al., 2009).

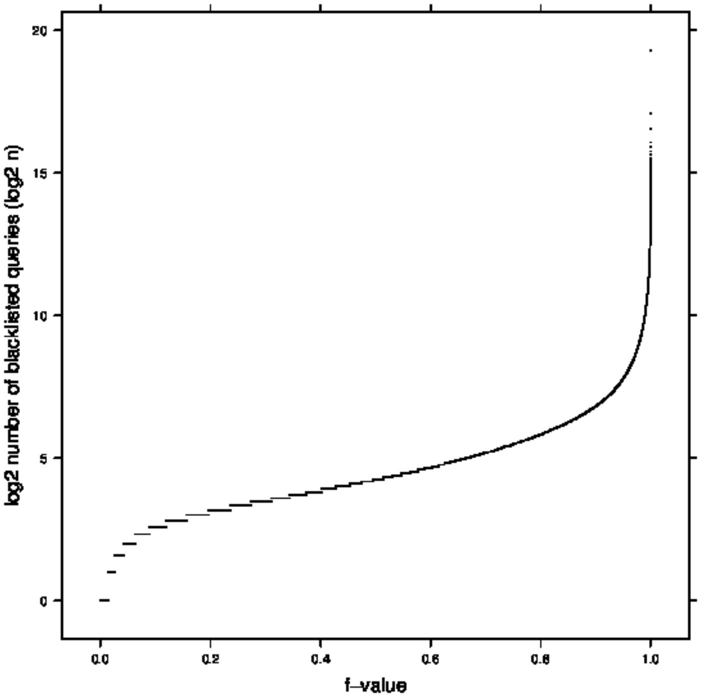

Figure 3.2.: Quantile-quantile plot of blacklist query counts

Figure 3.2 illustrates the visual recombination using the quantile-quantile plot. Here, on the Y axis the empirical blacklist query count quantiles are plotted against the uniform distribution on the X axis.

3.2.3 Analytic Recombination

Analytic recombination is simply a continued analysis of the previously divided or analysed data. Methods of analytic recombination are quite frequently applied to datasets. Here an example of analytic recombination pertaining to the Spamhaus DNS blacklist queries is seen.

The blacklist query count distribution reveals that the queries are distributed over a wide range of values. However, for further analysis a representative sample of the queried hosts along with their queries is needed.

The chosen number of samples is 500, equally spaced along the Spamhaus DNS blacklist query count distribution, but on a log 2 scale. These values are chosen from a range of values that best represent the data and provide adequate data points for the analysis. 16 (2^4) queries are established as the minimum number of queries required for the analysis. 18,000 ($2^{14.13}$) is chosen as the maximum number of queries as it is representative of the 99.9th percentile of the Spamhaus DNS blacklist query distribution. 500 values that are equally, or roughly equally[6] spaced between 16 and 18,000 are chosen as the representative sample.

This sample data is an analytic recombination of the original Spamhaus DNS blacklist query dataset. It is smaller in size compared to the original dataset and is used to analyse queries of hosts as a *Marked Point Process*. Every query of every 500-sampled host is considered as an event in time. The event or *mark* represented by each point might be different, for example, a blacklist or a not-blacklist.

3.3 D&R Computational Environment

D&R computational environment harnesses the capabilities of the R programming environment and the parallel computing framework - Hadoop. The environment is made up of three open source applications: R, Tessera (RHIPE), and Hadoop as illustrated in figure 3.3.

[6] The space between 2^4 and $2^{14.13}$ is equally divided into 500 points. Any point that does not have

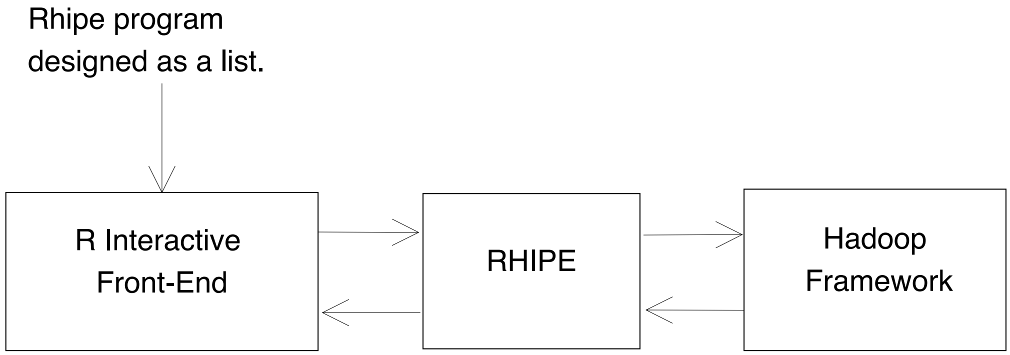

Figure 3.3.: D&R components

3.3.1 R Interactive Front-End

An analyst using the D&R computational environment interacts with the D&R framework through the R interactive front end. Programs that exploit the combined advantages of this environment are written in the R programming language. This program is designed as an R-*list* and can be divided into two parts. The first part is the actual R programming language code that implements the D&R framework programmatically. The second is the set of configurations that are required by R, Tessera (RHIPE) and Hadoop to run the D&R procedures successfully.

3.3.2 Tessera (RHIPE)

Tessera is the interface that binds the D&R environment. It interfaces the R interactive front-end with Hadoop. The analyst invokes the Hadoop framework through Tessera. Tessera uses the program submitted by the analyst to create R and Hadoop configuration files, and sets up the data for analysis.

a host with the exact number of queries is filled by the host with the closest number of queries.

Tessera implements the D&R framework using the MapReduce paradigm, and allocates specific map and reduce tasks from the analyst's program. It applies the R functions across the dataset in an embarrassingly parallel fashion.

There are three different outputs from the D&R framework: the divided subset data, the output of the functions applied on the subset in an embarrassingly parallel fashion, and the recombination of results. To begin with, the divided subsets and the recombination results are always written back to the HDFS. Output of the function applied in an embarrassingly parallel fashion is written to the HDFS as needed.

D&R exploits Hadoop effectively by using the MapReduce framework for its operations. Hadoop is designed to run a finite number of tasks on a virtually unlimited number of datasets. For this reason Hadoop schedules a set of tasks and sends subsets to them until all the subsets have been processed. Contrary to the familiar program execution where data is sent to the program, on Hadoop the task is sent to the node where the data resides. This approach minimises inter-node communication and speeds up the tasks and jobs.

3.3.3 Hadoop

Hadoop is another component of the D&R framework. It is a library and a framework that implements scalable distributed processing of large datasets using a cluster of heterogeneous, commodity computers. Each of these machines follow data locale computing, thus making the framework efficient. The framework also manages data and process redundancy (Bialecki et al., 2005).

The Hadoop framework is divided into two components, data storage and management, and computation. The data storage and management part of Hadoop is called Hadoop Distributed File System (HDFS). HDFS is a Java-based file system that is designed to be distributed, scalable, and elastic across many storage nodes.

A primary feature of this file system is data redundancy, as Hadoop is usually run on marginally reliable commodity hardware (Shvachko et al., 2010).

MapReduce is the computational aspect of Hadoop, which is an open source implementation of Google MapReduce developed internally by Google (Dean and Ghemawat, 2008). The MapReduce framework interprets the job and task configurations, runs tasks on subsets by applying analytic methods on the subsets, and recombines the results.

3.4 D&R Environment Implementation

The D&R framework is designed to be a robust analytical framework. The versatility of D&R framework comes from its ability to run on multiple platforms.

Current day computational platforms can be broadly classified into *bare-metal*[7] and *virtual*[8] platforms.

The *BoHD* analytical cluster, built specifically for working on research threads in cybersecurity, implements D&R on bare-metal machines. Implementing and maintaining an analytical cluster on bare-metal machines is time-consuming due to the limitations of installation and update automation. They are also expensive as there is a high cost for procuring hardware. Bare-metal clusters can take up heavy computing as all their hardware components can be maximized for performance.

Virtual machines have a low adaptation cost. Virtual machines are leased out by Amazon AWS[9], Google and Microsoft in units of time, and the cost per unit time is low and affordable. Virtual machines also have a quick, automated installation process, and do not need to be maintained. This keeps the time required for platform preparation negligible. They are also highly scalable. Factors like, low cost, negligible time investment and high scalability makes virtual

[7] Bare-metal platforms are a set of physical hardware components that can be used only when the right operating system, file system, applications, and programs are installed on it.

[8] Virtual machines are computer systems emulated by software or a combination of software and hardware.

[9] AWS stands for Amazon Web Services.

machines a very favourable platform for running large analytics, especially D&R. Virtual machines do lack the computing “punch” of a hardware-rich, well designed bare-metal cluster; but by scaling the number of virtual machines its computational power can match that of a bare-metal cluster.

D&R has been implemented on three virtual machine platforms - Amazon EC2, Vagrant, and Amazon EMR (Barthur, Ashrith and Hafen, Ryan, 2015; Barthur, Ashrith and Hafen, Ryan, 2014; Hafen, Ryan and Barthur, Ashrith, 2014). While, Amazon EC2 and Vagrant are purely virtual machine provisioning services, Amazon EMR provides Elastic MapReduce[10] along with bare-virtual machines.

[10] ElasticMapReduce is a Hadoop platform managed by Amazon and offered as a service.

CHAPTER 4. SPAMHAUS DATA - GENERIC ANALYSIS

A generic data analysis is critical for data-driven research. It helps understand the distribution of the data, check the assumptions, and identify biases. It also helps in charting an analytical path and identifying suitable models for the data (Seltman, 2012).

The Spamhaus dataset is collected for nine months. The data consists of 13.178 billion queries. These queries are divided into IP-address queries, accounting for 10.615 billion and 2.561 billion domain-address queries, respectively. There are 982,293 unique querying hosts that send out these 13.178 billion queries. The IP address queries consists of 207 million unique hosts[1] and accounts for the 10.615 billion queries.

The IP and domain address categories of queries can be further divided into queries with a blacklist response and queries without a response. There are 8.864 billion IP address queries, 2.476 billion domain address queries that have no response, 1.168 billion IP address queries and, 81 million domain address queries that have a blacklist result. The 1.168 billion queries that are blacklisted are for 59 million unique queried hosts.

Hosts can be blacklisted for multiple reasons, but for the purposes of analysis they are simply identified as blacklisted or not-blacklisted. An IP address host is considered blacklisted if it is listed in any one of the five IP address blacklist categories: spam, composite snowshoe spam, exploit, ISP policy violation, and Spamhaus policy violation. Similarly, a domain address is considered blacklisted if it is listed in any one of the two domain address blacklist categories: malicious domain and domain redirection.

[1] Unique hosts are classified by their IP addresses.

4.1 Generic Query Analysis

Generic data analysis consists of studying data represented by numerical and visual statistics. One common visual representation of data distribution is through quantile-quantile plots (qq-plots), which plot the quantiles of the data against a theoretical quantile, e.g. *Uniform* quantile.

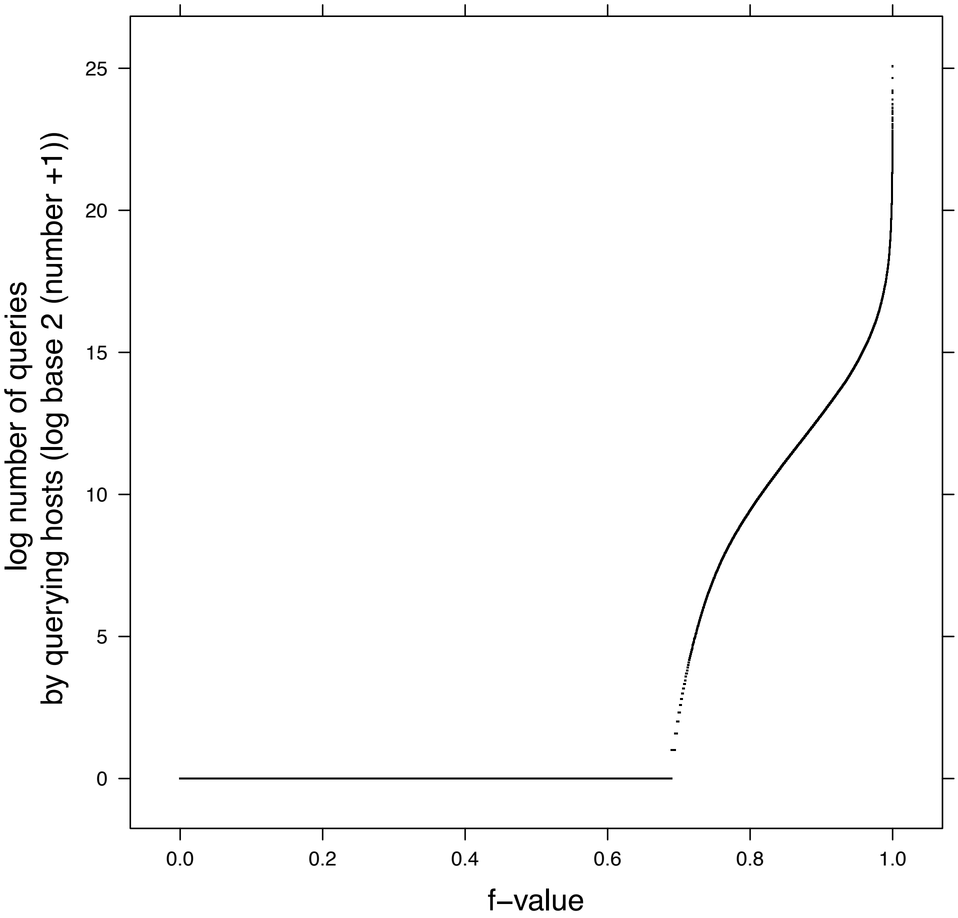

Figure 4.1.: Quantile - quantile plot of log2 number of queries by querying hosts

Figure 4.1 visually describes the distribution of the number of queries by querying hosts. The number of queries by querying hosts vary between one to 16 million. Due to the wide data range, the number of queries are transformed to a *log2* scale and plotted on the Y-axis.

The distribution of the queries by querying hosts is visibly skewed. 70% of the 982,293 querying hosts query only once over the nine month collection period.

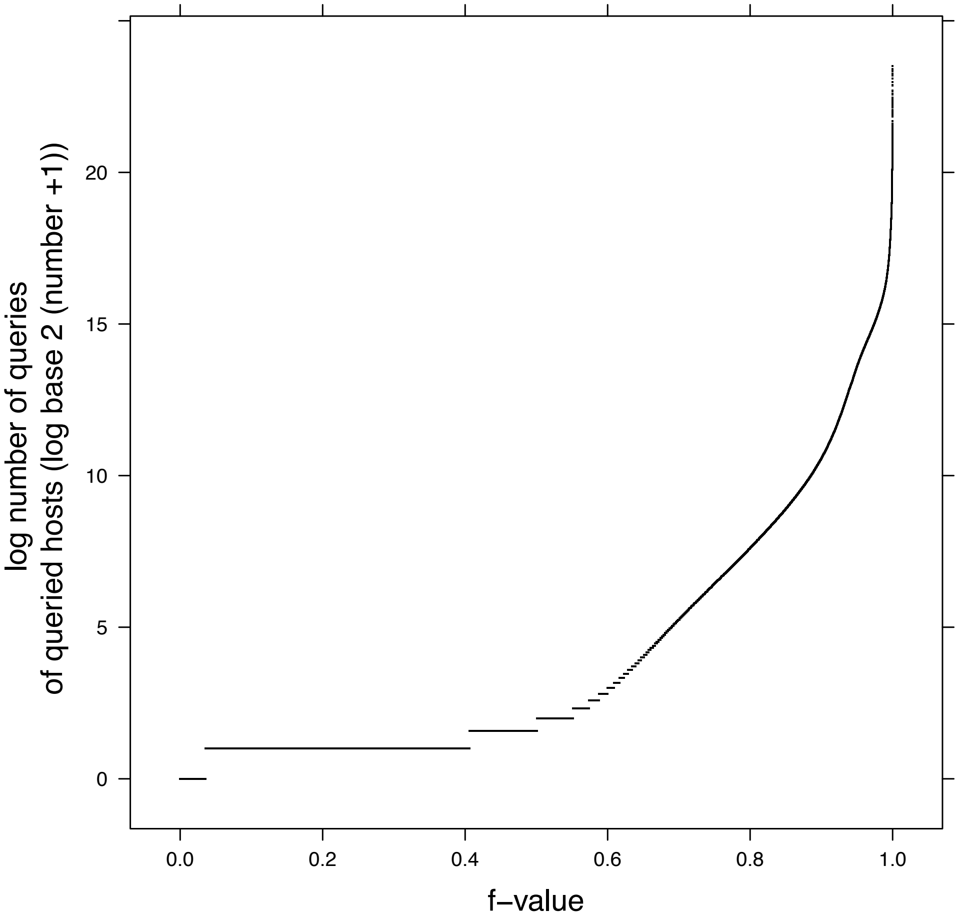

Figure 4.2.: Quantile - quantile plot of log2 number of queries of queried hosts

The distribution of the queries by queried hosts is visible in the qq-plot in the figure 4.2. The data points in this plot are scaled down by a log2 transformation for better interpretability, similar to qq-plot of queries by querying hosts. Here, about 50% of the queried hosts, out of 207 million are queried less than five times over the nine-month period. Similar to the distribution of the number of queries by querying hosts, the distribution of the number of queries for queried hosts is skewed. Compared to the query distribution by the querying hosts, the queried hosts show a much more gradual drop in density with the increase in number of queries.

4.2 Blacklisted Host Query Analysis

The Spamhaus blacklist query analysis research focuses primarily on studying only queries of hosts that are blacklisted, by time. The data for this study is carefully curated by three subset divisions using the D&R framework and stored in data structures optimised for the analysis.

A fundamental study that supports the blacklist query analysis is the general study of the properties of the queries. The query distribution by querying and queried hosts, blacklist query fractions and ratios, and query rates are properties that help understand the data. They also help identify data that are most rich for Analysis and help decide the focus area of study and the analysis process.

Figure 4.2 in the previous section visually describes the distribution of the queries by queried hosts. It is seen that 50% of the hosts in this population have less than five queries across nine months suggesting a limitation in data. Statistically modeling blacklist queries by host requires a minimum number of data points - queries per host. Therefore for all analyses of blacklist queries and blacklisted queried hosts, 16 is considered the minimum number of queries.

Figure 4.3 is a qq-plot that describes the distribution of the queries by queried hosts. This plot visually compares the quantiles of the number of queries by queried hosts against the theoretical quantiles of a uniform distribution, on the Y and X axes, respectively. The number of queries by queried host are plotted with a log2 transformation.

This qq-plot of the distribution of the queries by queried host is censored. It only considers hosts with 16 or more queries for analysis. It also shows a smoother

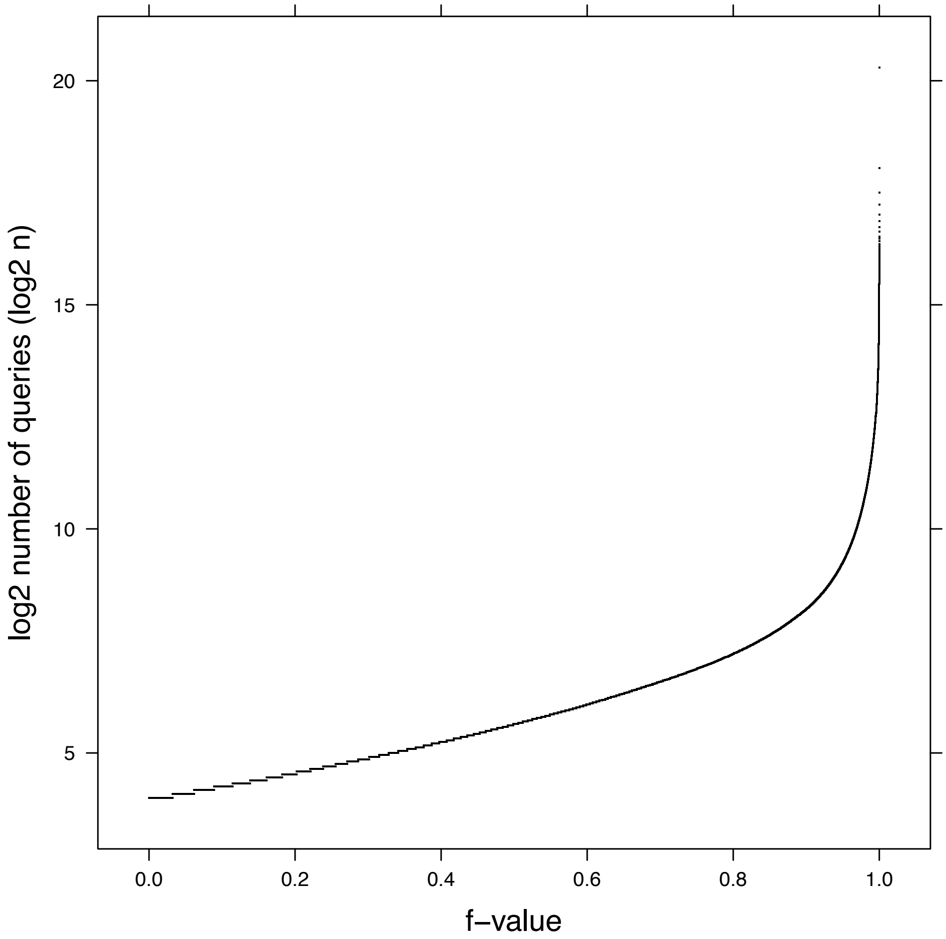

Figure 4.3.: Quantile - quantile plot of log2 number of queries of querying hosts

cumulative distribution gradient when juxtaposed and compared with the plot in figure 4.2. There is an inherent skewness in the data that is visible in the qq-plot.

The plot in figure 4.4 is a qq-plot of the distribution of the number of blacklist queries by host. The number of blacklist queries by host varies from one to approximately 500,000. The large range limits the visual perceptibility of the distribution. Therefore, similar to the plot in figure 4.3 the data is transformed to a log2 scale.

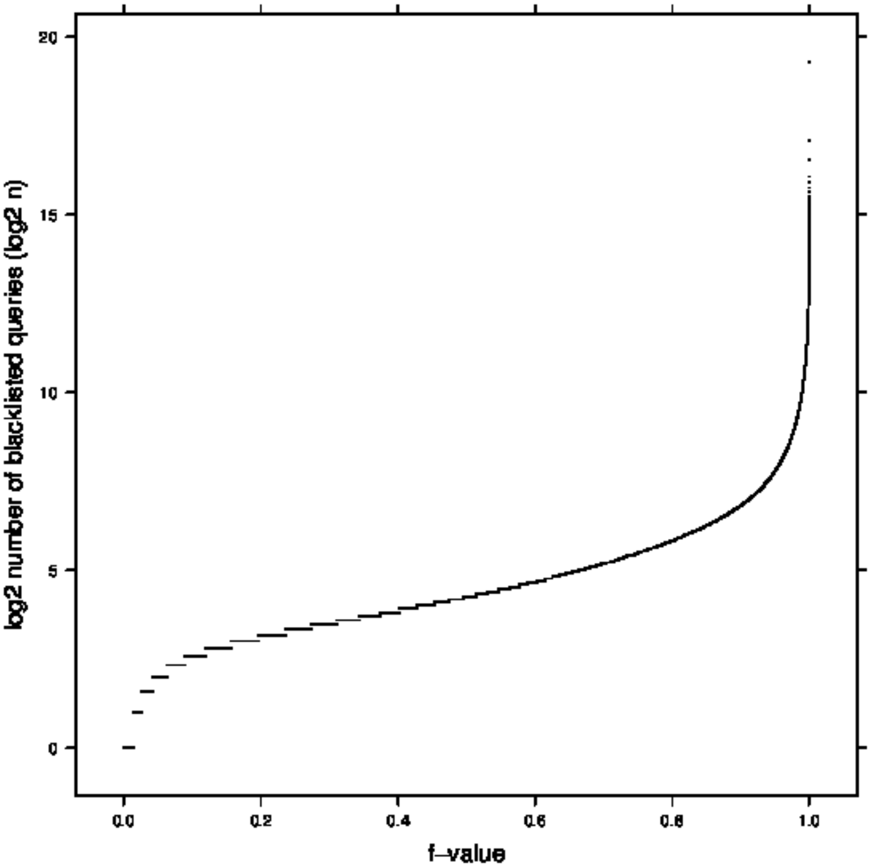

Figure 4.4.: Quantile - quantile plot of log2 number of blacklisted queries of querying hosts

The blacklist query distribution has a tail unlike the distribution of all queries by host. It is also dissimilar to distribution of the queries of hosts in figure 4.3 suggesting that all queries and blacklist queries show independence in behaviour.

The next set of qq-plots describe the distribution of the blacklist query fraction.

The figures 4.5 and 4.6 visually describe the distribution of logit2 blacklist query fraction against theoretical distributions, respectively. The distributions show

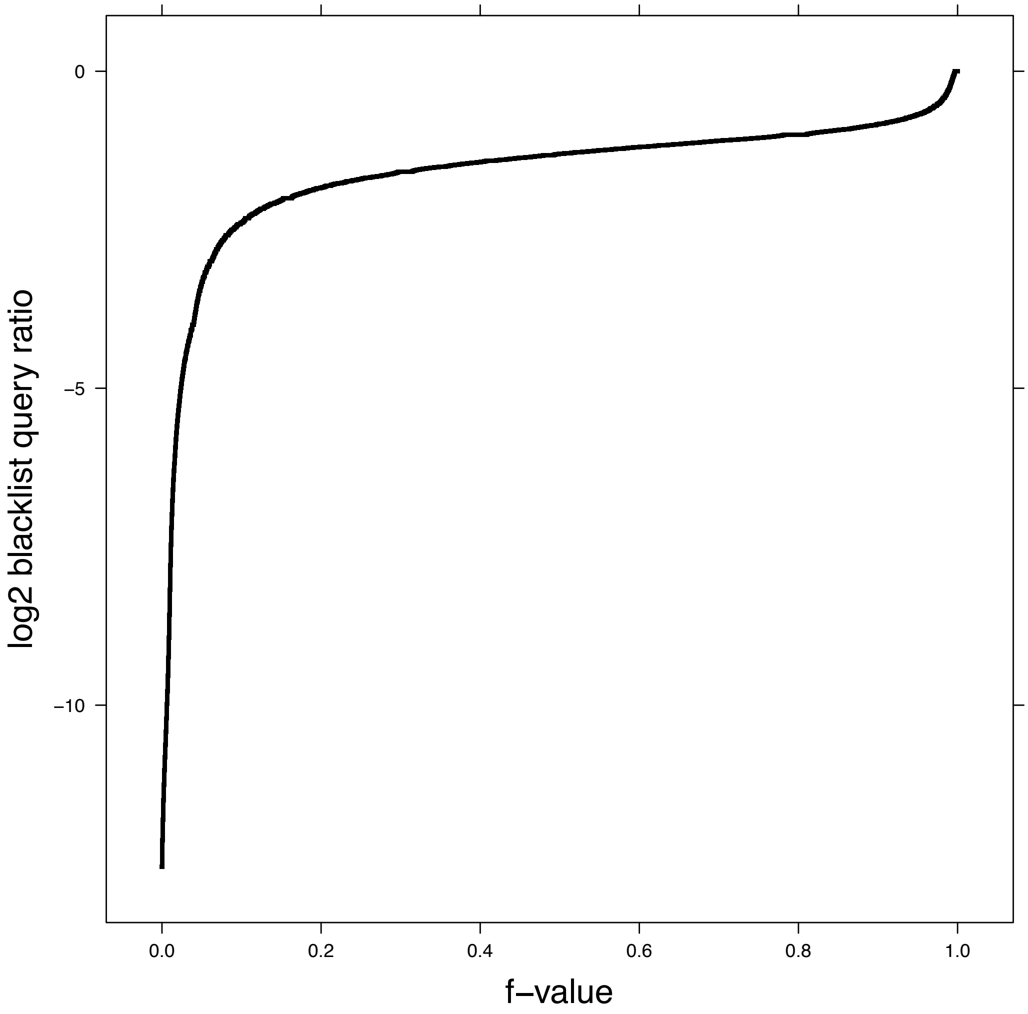

Figure 4.5.: Normal Quantile - quantile plot of logit2 blacklist query fractions

a long tail, with the probability density between $2^{-2.5}$ and 2^{-2}, which is between 17% to 25% of the queries by queried hosts that are blacklisted.

The qq-plot in the figure 4.7 characterises the distribution of the rate of queries per second by host. The plot shows that the query rate varies from 2^{-20} to 2^{5} and the distribution is gradual. This highlights that the queries and hosts collected in this study are well defined across the spectrum of query rate.

The blacklist query rate distribution is characterised in the figure 4.8. The qq-plot, similar to other distribution plots in this generic data analysis, uses a log2

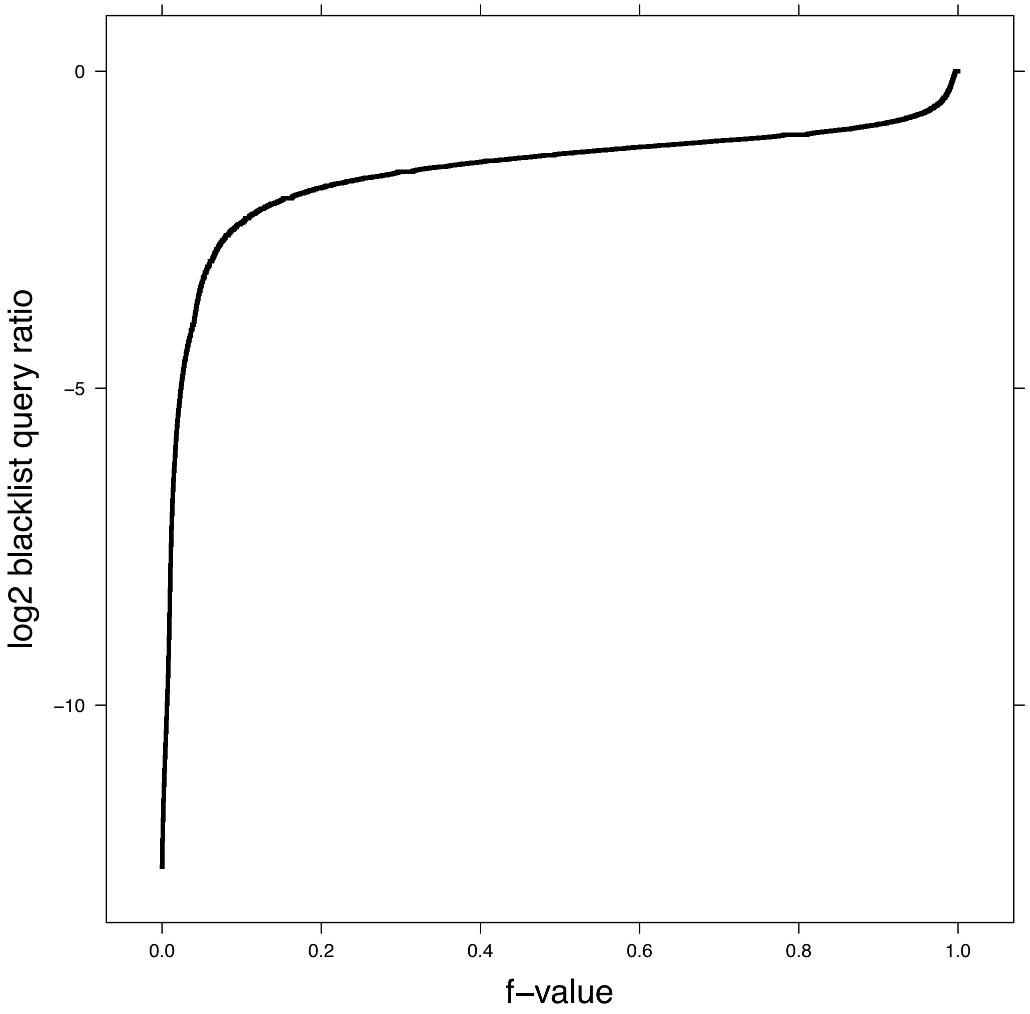

Figure 4.6.: Uniform Quantile - quantile plot of logit2 blacklist query fractions

transformation to make the data more perceivable. The log2 transformation helps a rather large data range - 2^{-23} to 2^{20} queries per second. Much of the probability distribution remains below 0 and between 2^{-20} and 2^{-5}. These query rates are similar to the observed all query rate. The range of the blacklist query rate is larger than the range of the all query rate, suggesting that some blacklist queries occur in bursts and in small-time intervals.

Properties like distributions, fractions, rates of queries, and hosts, are diagnostic in nature. They help in identifying areas of data that are rich in

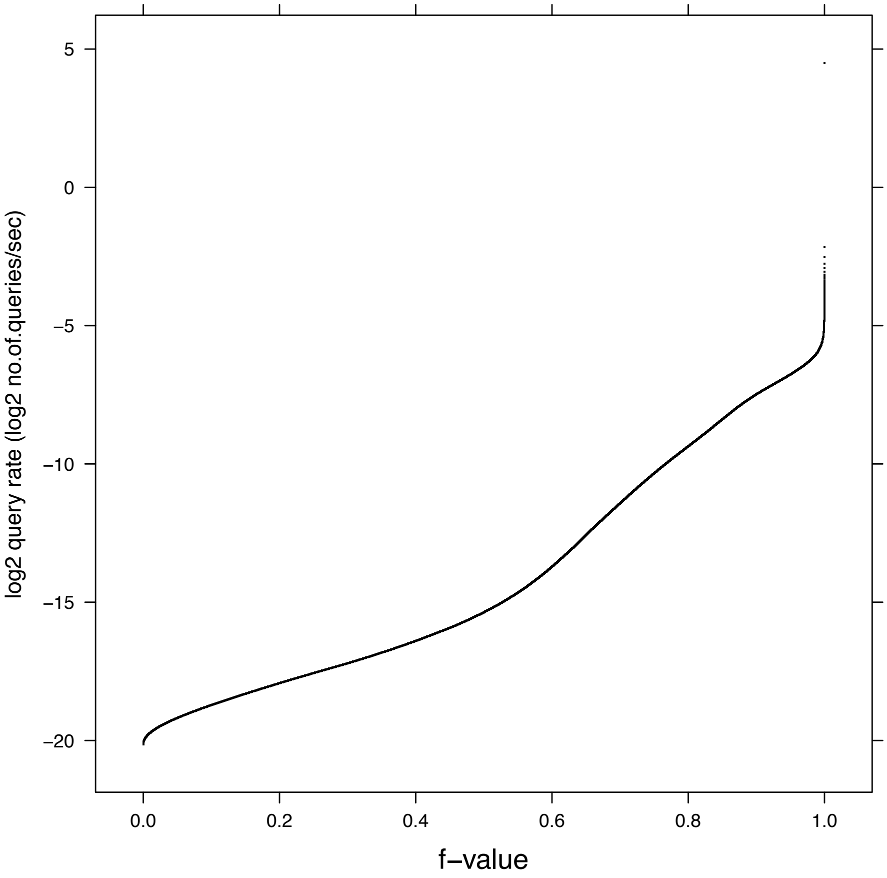

Figure 4.7.: Quantile - quantile plot of log2 number of queries per second

information. They also help in creating an analysis outline for the research study. In the next chapter, Marked Point Process (MPP) is used to study the queries, and hosts.

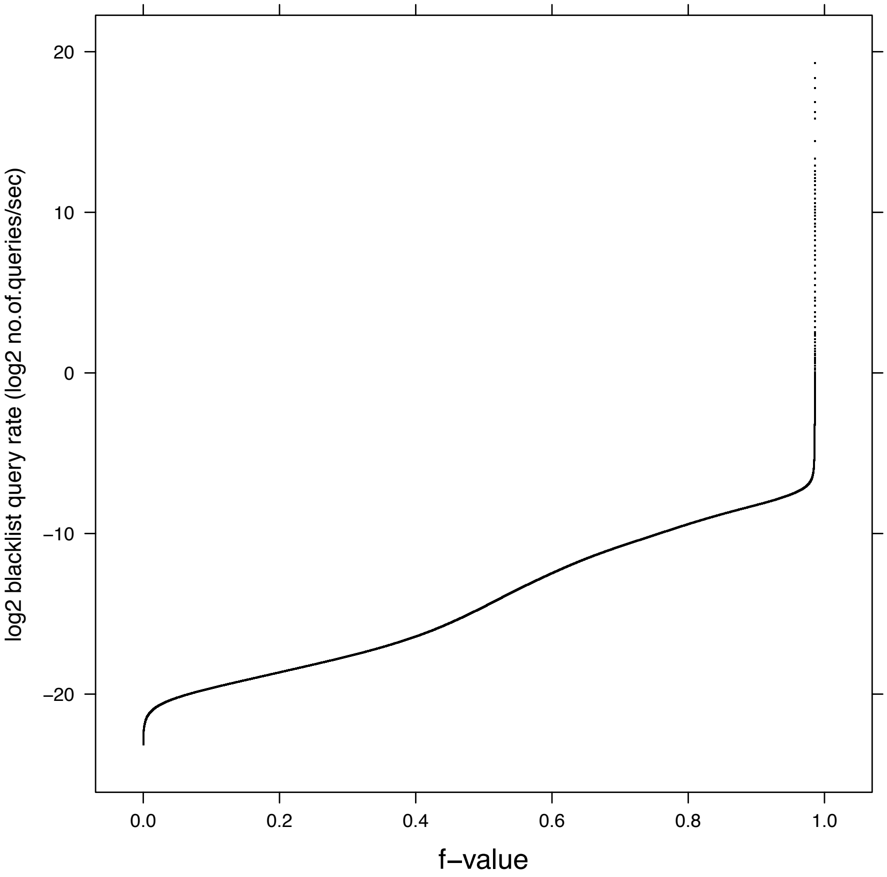

Figure 4.8.: Quantile - quantile plot of log2 number of blacklist queries per second

CHAPTER 5. SPAMHAUS MARKED POINT PROCESS ANALYSIS

A point process is a collection of random variables in a space. Each point in the space represents an event. The points could be points in time, geo-spaces, or any user defined space.

A marked point process is a point process where each event or point contains information in addition to the time or location of the event. For example, occurrence of earthquakes are identified as a point process by time and location. The magnitude of the earthquakes is additional information, additional to time and location of the occurrence (Schoenberg et al., 2002).

The Spamhaus blacklist query-response too are point processes in time - random set of events occurring in a time space. They are further classified as marked point processes as they carry additional information about - queried host, querying host, blacklisting, not-blacklisting, and category of blacklisting.

5.1 Representative Sample Analysis

A marked point process analysis of the queries first requires a representative sample of the data. The representative data must capture the distribution of the actual population for the analysis to be credible. Therefore a representative sample of the data by queried hosts needs to be constructed which represents the population fairly.

The number of queries by queried hosts has an extremely large range, starting from 2^4(chosen as the lower cut-off) to 2^{22}. 99.9th percentile of the actual population is considered to be the upper limit of the representative sample, this comes to about 18,000 queries. Hence, the new range for picking the representative samples is between 16 and 18,000 queries. The data range is still fairly large, therefore the samples are chosen on a log2 scale. 500 equally spaced unique values of number of queries from 16 to 18,000 are selected, and for each of these values a unique host that has the exact or proximate value of queries is chosen for the representative sample.

Link to viewgraph

Figure 5.1.: Normalized query time vs. number of events

The first plot of the representative sample analysis is the normalized time plot shown in figure 5.1.

The normalized time plot describes the relation between query events and the time of their occurrence for each of the 500 representative hosts. The query time range for each queried host is scaled down to 0 and 1, separately. And each individual query event is scaled down to a value between 0 and 1 based on its query time relative to the query time range and plotted as a point on the plot.

The normalized time plot in figure 5.1 has the normalized time on the Y axis and a scaled down value of the event number on the X axis. All plots primarily display a monotonic behaviour as successive query events occur at successive points in time, thus preserving the order. It is difficult to observe the monotonic behaviour on a few panels as the extreme values of query times distort the scale.

The plot identifies a blacklisted response as a red point on the plot and a not-blacklisted response in black.

It is seem from the plots that similar kinds of responses: a set of blacklist or not-blacklist, tend to occur together. The panel for each host shows that a set of blacklist, and set of not-blacklist responses occur in intervals across the entire query period.

The similar response intervals are visible across the representative data set. The occurrence of these intervals is advantageous to the analysis, as now, instead of individual queries the blacklist and not-blacklist intervals which are grouped queries, can be studied. These intervals are classified as *On* and *Off* events. An on event is a set of sequential responses that identify a host as blacklisted, and an off event is a set of sequential responses identifying a host as not-blacklisted.

5.1.1 On-Off Event Analysis with Representative Sample

The on-off events aggregate queries across hosts. These events seem to occur across the data very naturally. The phenomenon of similar-response queries is telling, and crucial for all future analyses on the Spamhaus blacklist query-response data.

Statistics, such as the duration of an event or the number of queries in an event, are deeply informative. They help one understand the behaviour of the queries-responses, and hosts.

Link to viewgraph

Figure 5.2.: Event Duration vs. Event Number

The figure 5.2 is a plot of the on-off event duration vs. the on-off event numbers. The on-off event duration is plotted on a log2 scale.

The figure 5.2 shows the duration of the events. The plots are arranged in an increasing order of the number of queries. Each panel represents a host out of the 500. The events range between 2^5 seconds to 2^{10} seconds for majority of hosts. With the increase in number of queries the event duration tends to stabilize between 2^8 seconds, and 2^{10} seconds. Having a majority of the event durations around 2^8 - 2^{10} suggests a strong tail behaviour. This can be confirmed by studying the distribution plots.

Link to viewgraph

Figure 5.3.: Uniform Quantile Plot of Off Event Duration

Link to viewgraph

Figure 5.4.: Uniform Quantile Plot of On Event Duration

The figures 5.3 and 5.4 are uniform quantile plots that visually describe the off and the on event distribution, respectively.

Similarly, the figures 5.5 and 5.6 are normal quantile plots that describe the off and the on distributions.

Link to viewgraph

Figure 5.5.: Normal Quantile Plot of Off Event Duration

Link to viewgraph

Figure 5.6.: Normal Quantile Plot of On Event Duration

The qq-plots for distribution in figures 5.3, 5.4, 5.5 and 5.6 indicate a heavy tailed behaviour of the event duration. This behaviour becomes more significant with the increase in the number of queries.

The Spamhaus blacklist queries are studied as true random events and represented as marked point process in a defined space. The observed heavy tail in figure 5.2, 5.3, 5.4, 5.5 and 5.6 indicates a departure from true randomness. It suggests a phenomenon, an induced behaviour in the events, thereby suggesting an induced behaviour in the responses.

The marked point process analysis of the 500 equally spaced queried hosts helps refine the Spamhaus blacklist queried host analysis. A significant outcome of this analysis is the discovery and use of *On* and *Off* events to study query and host behaviour leading to sound preliminary results.

The analysis also highlights the limitations of the 500 equally spaced queried hosts representative dataset. While strong behavioural indications e.g. heavy tail, was visible with hosts with large number of on and off events, the insights from hosts with small number of events were limited and inconclusive.

5.2 Queried Host On-Off Event Interval Analysis

The marked point process analysis of the 500 equally spaced queried hosts provides two specific guidances. One, the on-off event analysis becomes the central focus of research and analysis. Two, the 500 equally spaced queried host dataset, a representative dataset, gives way to a more focussed dataset of the 102 most queried hosts, which will be used for analyses going forward.

In general, an analysis is designed to gather preliminary results from a representative dataset and then gravitate towards a more focussed dataset for deeper, compelling results.

The 102 most queried hosts dataset contains Spamhaus queries and responses of a set of hosts who have been queried the most. The number of queries for the most queried host ranges from 140,00 to 1,286,000. Figure 5.7 characterizes the distribution of the number of queries for the most queried hosts. 85% percentile of data is below 400,000 queries with the remaining 15% percentile making up the larger values going up until 1,286,000.

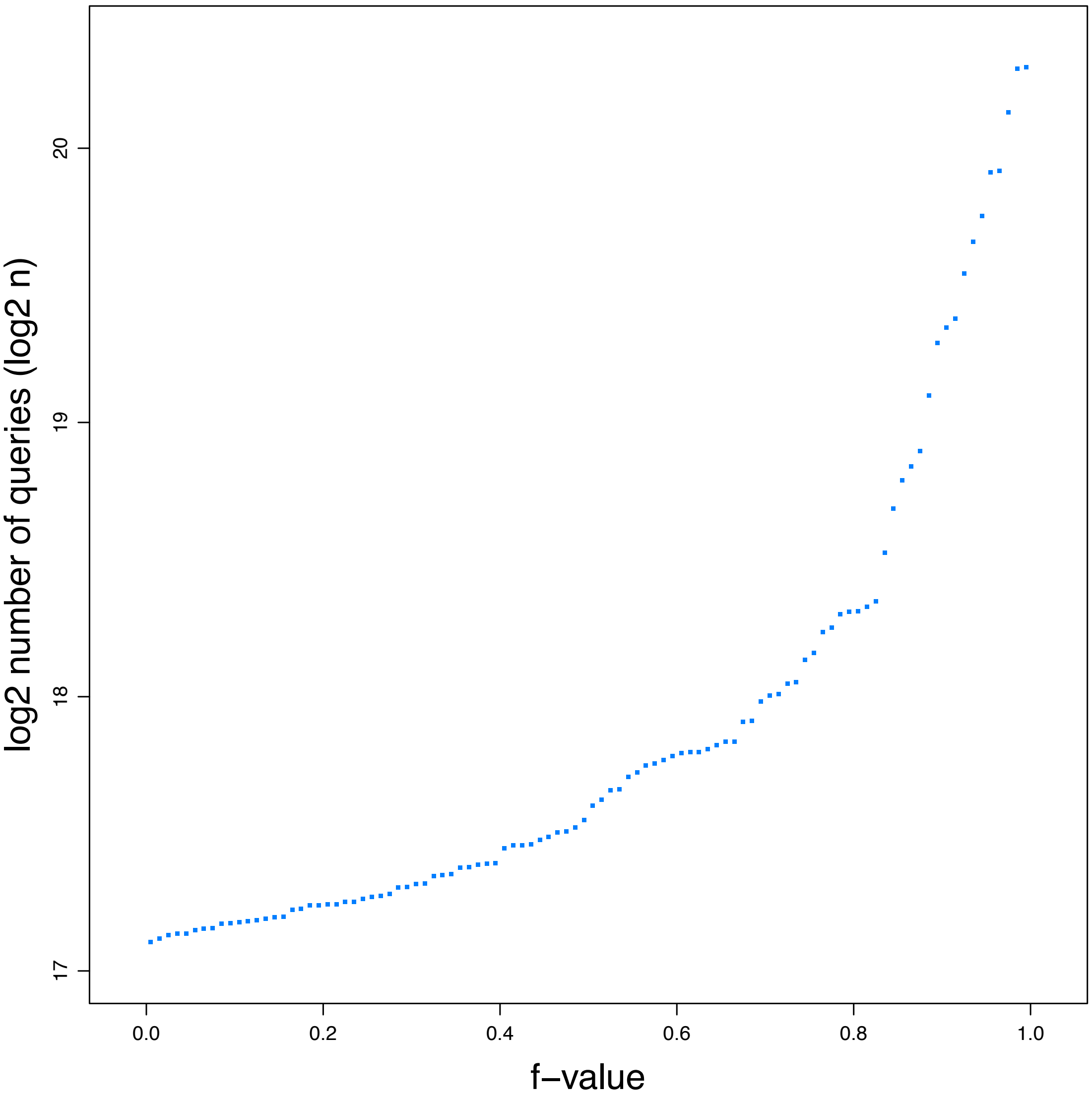

Figure 5.7.: Quantile - quantile plot of log2 number of most queried hosts

5.2.1 On-Off and Query Relation Analysis

The marked point process analysis of the 500 equally spaced queried hosts highlights a relation between the On-Off events and the number of queries. This relationship is further studied by using a more focus dataset - a set of 102 highly queried hosts.

The analysis studies the relation between the events and the queries by plotting the rate of events against the rate of queries for each of 102 highly queried hosts. Further, the points on the plot are fitted with a regression model and the coefficients of the regression model are used as a measure of fit. They are also used for studying the relation between On-Off interval rate and query rate.

The On-Off interval rate and the query rate are calculated using the elapsed time of queries for an individual host. This elapsed time is measured as the difference between the time of the first query and time of the last query for a specific host. The elapsed time is now divided into 100 equal time intervals. The number of On-Off intervals and the number of queries in each part are computed and divided by the length of the time interval, in seconds. The computation gives the On-Off interval rate and the query occurrence rate which is then converted to rates per day.

The query rate and the corresponding On-Off interval rate are plotted on a log2 scale, on the X and the Y axes, respectively. The relationship between the On-Off Interval rate and the query rate can be seen in figure 5.8.

The points on the plot are fitted using a robust regression linear model to account for any influential outliers. And the line of fit for the robust regression linear model is drawn on the plot. The plot in figure 5.8 is ordered by the number of queries.

Link to viewgraph

Figure 5.8.: Log2 On-Off intervals per day vs. Log2 number of queries per day

The plot in the figure 5.8 displays a log-linear relationship between the On-Off interval rate and the query rate. An increase in the log2 query rate corresponds to an increase in the log2 On-Off interval rate. The regression line shows a good fit for the 102 hosts.

The relation between the On-Off interval rate and the query rate is studied further by analysing the slope of the regression. Figure 5.9 demonstrates the quantile-quantile distribution of the slopes of the linear robust regression.

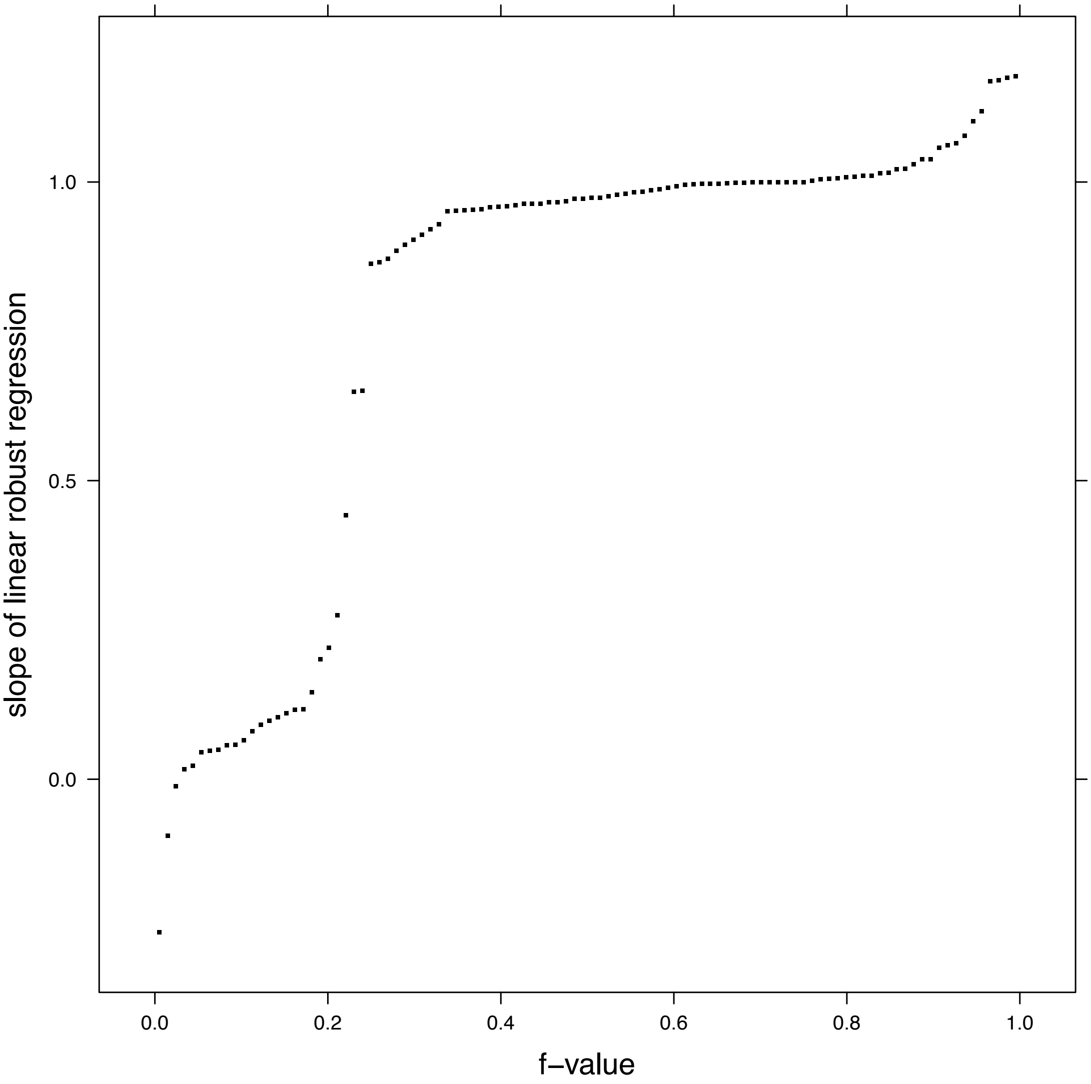

Figure 5.9.: Quantile - quantile plot of slope of linear robust regression

The quantile-quantile distribution plot of the slopes of the linear robust regression highlights that a large number of the slope values, about 60% percentile remain close to 1. This is a considerable number of values that show log-linear relation between query rate and the On-Off interval rate.

5.2.2 On-Off and Query Relation Analysis - Spam Queries Only

The On-Off interval rate vs. the query rate analysis using the Spamhaus blacklist data for the 102 highly queried hosts contains responses for five different blacklisting types: spam, composite snow shoes spam, exploit, Spamhaus policy violation, and ISP policy violation. Therefore, as a part of the On-Off and query relation analysis, it is important to verify that the log-linear relation between the On-Off interval rate and the query rate is a phenomenon of the underlying, individual blacklisting categories and not just the aggregated blacklisting[1].

The spam blacklisting is chosen as a candidate blacklist type to verify the log-linear relation as it is the most popular blacklist service provided by Spamhaus. The On-Off and query relation analysis on the aggregated blacklist data is replicated on the spam only blacklist data.

The figure 5.10 characterizes the relation between On-Off interval rate and the query rate for spam only, for each host.

Link to viewgraph

Figure 5.10.: Log2 On-Off intervals per day vs. Log2 number of queries per day for spam only

In the On-Off interval rate and query rate relation for spam-only analysis, the total number of queries remain the same as the On-Off interval rate and query rate relation for aggregated blacklisting. The number of On-Off events vary because

[1] For the entire analysis blacklisting is an aggregated blacklisting of all types - spam, composite snow shoes spam, exploit, Spamhaus policy violation, and ISP policy violation.

spam is the only blacklisting category considered for this analysis. This shift is visible when comparing figures 5.10 and 5.8, as the X-axes between them remain the same, but the Y-axes differ.

The plot in the figure 5.10 displays a log-linear relationship between the On-Off interval rate and the query rate. This relationship is similar to the relationship between On-Off interval rate and query rate in figure 5.8.

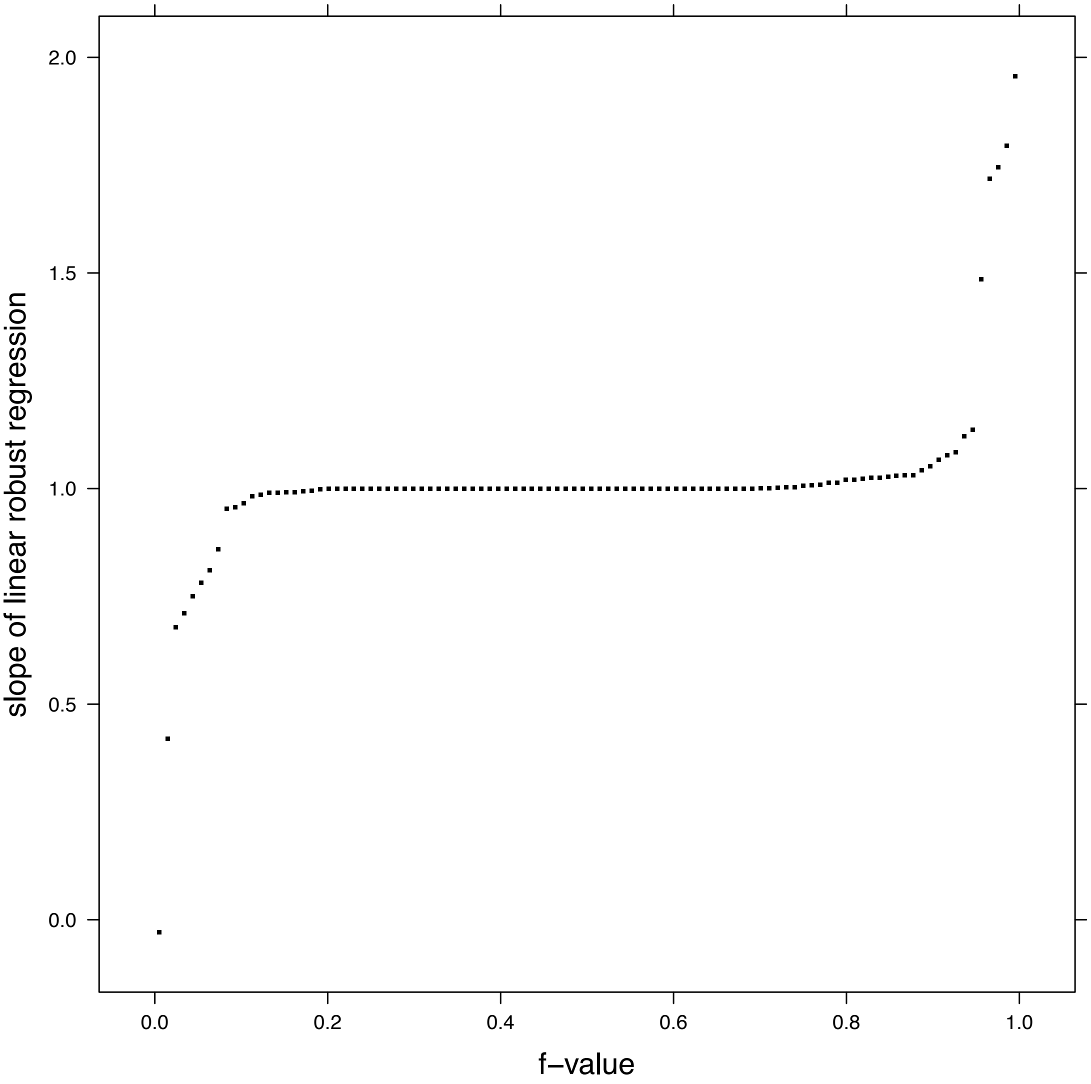

Figure 5.11.: Quantile - quantile plot of slope of linear robust regression for spam only

The figure 5.11 represents the distribution of the slopes of the linear robust regression. About 80% percentile of the slope values are at 1. The distribution plot along with the On-Off interval rate vs. query rate plot highlight the existence of a log-linear relation between the On-Off interval rate and the query rate.

The On-Off interval rate and query rate relation analysis bring about a visible relation between the On-Off events and number of queries. Further, this relation is indicative of an underlying phenomenon of the Spamhaus blacklist response, which seems to be an anomaly.

5.3 Event Run Analysis

The 102 most queried host dataset is fundamental to the On-Off interval and query rate relation analyses. In both these analyses the queries are clustered into sequentially identical blacklist and not-blacklist groups called On and Off events, and the durations and rates of these events are studied. In the event run[2] analysis, the behaviour of queries that form an event run are studied.

5.3.1 On Event Run Analysis

The On event run analysis only studies the behaviour of queries in the On event runs. Based on the observations of this analysis, it is further expanded into fixed-sized query run analysis of all queries.

The On event run analysis utilized the 102 highest queried host dataset. This data is first broken down into On and Off events by host. Each of the On and Off events are separately aggregated by their run lengths[3]. The number of events by each run length are counted and stored as a *dataframe*. The data is filtered to remove all run lengths of Off events. Data of one host is excluded from the analysis

[2] A run refers specifically to the sequence of queries that form an On or Off event.

[3] A run length is the number of same-type queries - blacklist or not-blacklist, in an unbroken sequence.

as the host does not have the requisite number of events for the analysis, therefore the dataset is now reduced to the 101 highest queried hosts.

Link to viewgraph

Figure 5.12.: Quantile-quantile plot of log2 number of queries in an On event run

Figure 5.12 describes the distribution of the On event run lengths by hosts. The panels are ordered in an increasing order of events as shown in the strip label. A log2 transformation is applied to the run lengths for better discernibility.

The plots in figure 5.12 highlight skewness in the data. The plots also highlight that much of the distribution is populated with small sized event runs. This shows that On events largely have small run lengths.

The observable skewness in the figure 5.12 prompts a further study of the run length distribution. For this study the empirical probability of each run length is calculated. The empirical probability of a run length is given by the number of On events for a specific run length divided by the total number of On events for the given host. Additionally, an overall estimated model probability is calculated by dividing the number of blacklisted query-responses divided by the total number of query-responses for a given host. These two variables are used to study the run length distribution.

The plot in figure 5.13 characterizes the relation between the empirical probability and estimated model probability for each host. The empirical probability represented on the Y-axis is transformed with a negative log function (-log2). The estimated model probability is also transformed with a negative log function (-log2) and multiplied with the corresponding run length. The line `x=y` is represented on each panel.

Link to viewgraph

Figure 5.13.: Empirical Probability vs. Estimated Model Probability

A few hosts shown in figure 5.13 do not display a log-linear relation between the variables. Apart from those few that have a strong departure from a log-linear relation, a majority of the hosts do show a strong log-linear relation.

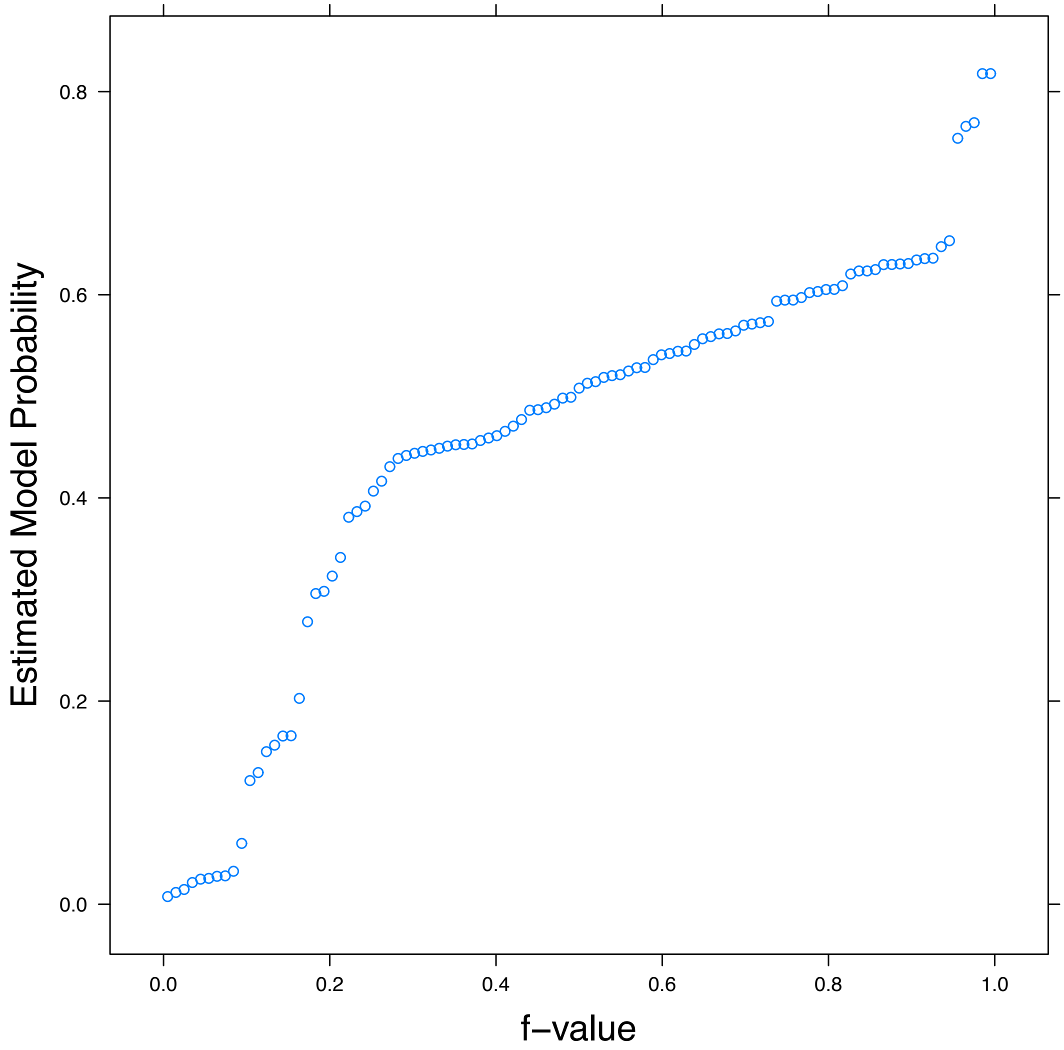

Figure 5.14.: Quantile - quantile plot of estimated model probability

Figure 5.14 describes the distribution of the estimated model probability. It is clearly observable that about 70% percentile of the values are between the value of 0.4 and 0.6.

5.3.2 Fixed Sized Query Run Analysis

The On event run analysis studies the query-response blacklist behaviour by utilizing the run lengths of On events. This is essentially an analysis of blacklist queries by variable length. In the fixed-sized query run analysis the query-response behaviour is analysed by partitioning the queries of a hosts into fixed-size runs. Unlike the On event run analysis, this analysis includes blacklist and not-blacklist query-responses; the fraction of the blacklist to the run length forms the fundamental variable upon which the blacklist behaviour is modeled.

The 102 highest queried hosts are used as the dataset in the fixed sized query run analysis. The queries-responses of each host up until the first blacklist response are disregarded for this analysis. The rest of the query-responses are divided into fixed-sized intervals of 256 query-responses in each interval. Two hosts do not meet the minimum number of query-responses required for this analysis and are therefore ignored for this analysis, thereby reducing the number of hosts that are analysed to 100.

Link to viewgraph

Figure 5.15.: Number of Blacklist Queries vs. Run Index Number

The figure 5.15 visually describes the variations in the number of blacklists across the 256 fixed query intervals for each host. The plot is ordered by hosts with increasing order of intervals.The number of intervals are identified in the strip label of the plot.

The consistent variation in the number of blacklist query-responses is highly visible in the plots. These variations can also be perceived with the progression of time as the intervals in the figure 5.15 are in time order.

Link to viewgraph

Figure 5.16.: Quantile - quantile plot of blacklist queries in 256 query intervals

The variations in the number of blacklists by hosts indicate that the query-response blacklisting could be a random process. To further investigate this, quantile plots of the blacklists are plotted in the figure 5.16. The plots in this figure are ordered by hosts, with increasing number of 256 query-response intervals. The plot shows that most of the hosts display a tail in their distribution. The blacklist query-response distribution across most hosts seems binomial in nature.

The binomial nature of the query-response is studied further by comparing the blacklist query-response fractions for each interval to a binomial distribution. The number of occurrences of each blacklist count in the 256 query-response intervals are recorded. Each of these counts are further divided by the total number of blacklist counts per host. Further, this data is used in the plots in figure 5.16.

An overall blacklist fraction for a host is calculated by the total number of blacklist query-responses divided by the total number of queries. Using the overall blacklist fraction, the binomial density is calculated for each visible count of blacklist query-response for a given host.

The figure 5.17 visually describes the fractions of blacklist and binomial distribution against the number of the blacklist query-response in an interval. Here, the density of the binomial distribution is plotted on the Y axis and the corresponding blacklist query-response size in that interval is plotted on the X axis. Additionally, the plot is overlaid with the fraction of blacklist intervals on the Y axis.

Link to viewgraph

Figure 5.17.: Fraction of Blacklist and Binomial Distribution vs. Number of Blacklisted queries in an Interval of size 256

The plot in the figure 5.17 is used to visualize the distribution of the fraction of blacklist. This plot shows that that data is fairly binomial in nature, but it is not convincing. The fit is studied by changing the size of the interval.

Link to viewgraph

Figure 5.18.: Fraction of Blacklist and Binomial Distribution vs. Number of Blacklisted queries in an Interval of size 128

Link to viewgraph

Figure 5.19.: Fraction of Blacklist and Binomial Distribution vs. Number of Blacklisted queries in an Interval of size 64

Link to viewgraph

Figure 5.20.: Fraction of Blacklist and Binomial Distribution vs. Number of Blacklisted queries in an Interval of size 32

Link to viewgraph

Figure 5.21.: Fraction of Blacklist and Binomial Distribution vs. Number of Blacklisted queries in an Interval of size 16

The next set of plots in figures 5.18, 5.19, 5.20, and 5.21 are decreasing in interval size - 128, 64, 32, and 16, respectively. These plots progressively show the fit of the fraction of blacklist to a binomial distribution.

The plot 5.21 shows the best fit of the fraction of blacklist, highlighting that a smaller interval size of 16 is supportive of the fraction of blacklist distribution, and is binomial in nature.

CHAPTER 6. CONCLUSION

The Spamhaus blacklist query-response analyses lead to a set of conclusions. Although they are preliminary in nature, they are foundational for further research.

The queried host analysis focuses solely on how query-responses vary by queried hosts. The foremost study of this analysis is the representative sample analysis. The representative sample analysis provides the "first look" of the data used for analysis.

The analysis helps in identifying and grouping the data as *On* and *Off* events which are the cornerstones of this research. The heavy tailed behaviour of the On and Off intervals helps recognise consistent, sub 1000 second On and Off event time periods. The plots in the figures 5.2, 5.3, 5.4, 5.5, and 5.6 visually identify the heavy tail and the consistent event intervals.

Any documented evidence of query-response behaviour or blacklist algorithmic implementation that would explain the consistent On and Off interval lengths are absent. Spamhaus has documentation on the lengths of time a host is blacklisted for different categories of blacklisting. Spamming IP addresses are blacklisted for six month periods (Spamhaus, 2016f). If the IP address belongs to professional spammers, or has repeatedly shown malicious behaviour, then the IP address gets blacklisted for years (Spamhaus, 2016f). The hosts that are identified under Composite Snowshoes Spam (CSS) have a small blacklisting period of two to three days due to their transient behaviour (Spamhaus, 2016f). The blacklisting time periods adopted by Spamhaus are far from the observed On-Off event time intervals. They are therefore the least likely candidates to explain the interval behaviour.

Spamhaus rebuilds its blacklist databases to fix blacklist errors. As Spamhaus uses the DNS protocol for communication the rebuilding of databases are seen as DNS zone updates. Each zone, representing a specific blacklist category is updated independently. The Spamhaus BlockList (SBL) - that identifies spamming hosts is updated every 10 minutes (Spamhaus, 2016f). The Spamhaus eXploit BlockList (XBL) and Spamhaus Policy BlockList (PBL) are updated every 15 minutes (Spamhaus, 2016a; Spamhaus, 2016d).

These zone update periods are remarkably close to the observed On-Off interval time periods. Therefore, it can be speculated, that the consistent event intervals are related to the Spamhaus zone updates. The periodic zone updates fix blacklisting errors, and with speculation, it could be said that the hosts in the study were blacklisted in error. But it is less likely that the entire representative sample dataset was repeatedly blacklisted in error. These speculative assessments suggest that the consistent interval behaviour to be an anomaly of the Spamhaus blacklisting.

The 102 highest queried host sample dataset replaces the representative sample as the dataset for analysis. This focused sample supports more intensive On-Off event analysis.

The plot in figure 5.8 demonstrates the relationship between the On-Off event interval rate and the query rate. The plot shows a perfectly log-linear relationship between the On-Off event interval rate and the query rate. The log-linear relationship conveys an increase in the event rate with an increase in the query rate.

The On-Off intervals merely represent sequential blacklist or not-blacklist query-response chains. The blacklist or not-blacklist is of a host and not an individual query. Therefore, theoretically the On-Off interval rate must be independent of the queries rate. But empirically it is observed that there is a clear correlation. This can be attributed to a phenomenon of queried host blacklisting that is occurring faster than the data capture rate of the experiment.

The fixed-sized query-response analysis is an important analysis of the queried host blacklist study. Here, query-responses for the highest queried hosts are divided into fixed size intervals and blacklist fractions in each of these intervals are studied. The plots in figure 5.16 reveal that the blacklist fraction distribution is binomial in nature. The blacklist could be observed as a random process instead of a deterministic, host-dependent event.

The fractions are analysed by plotting the blacklist fraction against the binomial distribution for a host. This is shown in figure 5.17. The plots in this figure show that the fractions are binomially distributed. A further variation in the interval length from 256 to 128, 64, 32, and 16 reveals a better fit of the fraction to a binomial distribution. The binomial fit for different interval lengths are shown in the figures 5.18, 5.19, 5.20, and 5.21.

The binomial distribution of the blacklist fractions indicates an underlying phenomenon by observation. This phenomenon is similar to a random process - *"a query-response coin flip"*. The blacklist fractions follow the probabilities of a coin flip in multiple trials, very closely. The plots in figure 5.15 and the relationship between the event rate and the query rate suggest a drift in the binomial nature of the blacklist. Therefore the randomness might be systemic to the Spamhaus blacklist itself or might be a statistical phenomenon. Future research in this space might be able to throw more light in identifying this phenomenon.

The deep and complex analysis of the Spamhaus blacklist query dataset has been possible due to the design of *BoHD* - an environmental designed for cybersecurity that is capable of processing and analysing network packets. The environment implements Tessera: the fundamental package for analysing large complex datasets.

BoHD and Tessera have both been responsible for the deep analysis on the Spamhaus data, as they enable an analyst to virtually analyse any size data by seamlessly and transparently implementing statistic, visualization, and analytic functions on the dataset.

The successful results of the Spamhaus blacklist dataset are testaments to the strengths and capabilities of BoHD.

CHAPTER 7. SPAMHAUS COLLECTION SUMMARY STATISTICS

7.1 Spamhaus Collection Summary Statistics - 6 Weeks

Category	Statistic
Collection duration	6 weeks
All queries	2,051,268,078
IP address queries	1,679,915,476
IP address queries with blacklisted result	331,193,986
Domain name queries	370,958,042
Domain name queries with blacklist result	13,093,693
Queries with no available response	1,657,054,290
Whitelisted queries	15

Table 7.1: Spamhaus Collection Summary Statistics (Barthur, Ashrith and Cleveland, William S. and Gerth, John, 2014)

7.2 Spamhaus Collection Summary Statistics - 31 Weeks

Category	Statistic
Collection duration	31 weeks
All queries	13,178,080,366
IP address queries	10,615,054,608
IP address queries with blacklist result	1,168,586,415
IP address queries with no available response	8,864,023,024
Domain name queries	2,561,462,755
Domain name queries with blacklist result	81,321,391
Domain name queries with no available response	2,476,098,548
Querying IP addresses	982,293
Queried IP addresses	207,081,108
Queried IP addresses with atleast one blacklist result	59,435,635

Table 7.2: Spamhaus Collection Summary Statistics (Barthur, Ashrith and Cleveland, William S. and Gerth, John, 2015)

LIST OF REFERENCES

LIST OF REFERENCES

Apache Hadoop (2009). SequenceFile.

Barthur, Ashrith (2014). `https://github.com/ashrith/dpkt`.

Barthur, Ashrith and Cleveland, William S. and Gerth, John (2014). Divide & Recombine for Big Data Analysis for Cybersecurity: Application on DNS Blacklist Query Study. CERIAS,2014. Presented at the 15th Annual Information Security Symposium.

Barthur, Ashrith and Cleveland, William S. and Gerth, John (2015). DNS Blacklist Query Study - A Big Data Approach. CERIAS,2015. Presented at the 16th Annual Information Security Symposium.

Barthur, Ashrith and Crabill, Doug and Tong, Xiaosu (2015). `https://github.com/tesseradata/docs-install-cluster`.

Barthur, Ashrith and Hafen, Ryan (2014). `https://github.com/tesseradata/install-vagrant`.

Barthur, Ashrith and Hafen, Ryan (2015). `https://github.com/tesseradata/install-whirr`.

Bialecki, A., Cafarella, M., Cutting, D., and OMalley, O. (2005). Hadoop: a framework for running applications on large clusters built of commodity hardware. *Wiki at http://lucene. apache. org/hadoop*, 11.

Brisco, Thomas (1995). RFC 1794: DNS Support for Load Balancing.

Cleveland, W. (1993). *Visualizing Data*. At&T Bell Laboratories.

Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113.

Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B., and Cleveland, W. S. (2012a). Large complex data: divide and recombine (d&r) with rhipe. *Stat*, 1(1):53–67.

Guha, S., Hafen, R., Rounds, J., Xia, J., Li, J., Xi, B., and Cleveland, W. S. (2012b). Large complex data: divide and recombine (d&r) with rhipe. *Stat*, 1(1):53–67.

Guha, S., Hafen, R. P., Kidwell, P., and S.Cleveland, W. (2009). Visualization Databases for the Analysis of Large Complex Datasets. *Journal of Machine Learning Research*, 5:193–200.

Hafen, Ryan and Barthur, Ashrith (2014). `https://github.com/tesseradata/install-emr`.

Hornik, K. (2016). R FAQ.

John Gerth (2011). Packet Collection Framework.

Krochmal, M. and Cheshire, S. (2015). Special-Use Domain Names. RFC 6761.

Levine, John (2010). RFC 5782: DNS Blacklists and Whitelists.

Li, Jianfu and Barthur, Ashrith and Cleveland, William S. (2012). Divide & Recombine Logistic Regression Testing. Technical report, Department of Statistics, Purdue University.

Messina, P. C., Williams, R. D., and Fox, G. C. (1994). *Parallel computing works !*, chapter Independent Parallelism. Parallel processing scientific computing. Morgan Kaufmann, San Francisco, CA.

Mockapetris, Paul (1987). RFC 1035: Domain Names-Implementation and Specifications. (1035).

Postel, Jon (1980). RFC 768: User Datagram Protocol.

Postel, Jon (1981). RFC 791: Internet protocol.

Postel, Jon (1984). RFC 895: Transmission of IP Datagrams over Experimental Ethernet Networks.

R Core Team (2016). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria.

Reynolds, J. and Postel, J. (1990). Assigned numbers. RFC 1060, Internet Engineering Task Force.

Schoenberg, F. P., Brillinger, D. R., and Guttorp, P. (2002). Point processes, spatial-temporal. *Encyclopedia of environmetrics.*

Seltman, H. J. (2012). Experimental design and analysis. *Online at: http://www. stat. cmu. edu/, hseltman/309/Book/Book. pdf.*

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA. IEEE Computer Society.

Spamhaus (2016a). Exploits Block List.

Spamhaus (2016b). The Domain Block List.

Spamhaus (2016c). The Domain Block List.

Spamhaus (2016d). The Policy Block List.

Spamhaus (2016e). The Spamhaus Block List.

Spamhaus (2016f). The Spamhaus Block List.

TCPDump Working Group (2014a). LibPcap - Packet Collection Library.

TCPDump Working Group (2014b). TCPDump - Packet Collection Tool.

TesseraIO Research Group (2015). `www.tessera.io`.

The Spamhaus Project Limited (2014). `www.spamhaus.org`.

Vixie, Paul (2000). RBL Usage.

White, T. (2009). *Hadoop: The Definitive Guide*. O'Reilly, first edition edition.

VITA

VITA

Ashrith Barthur was born in Bangalore, in 1983. He earned a Bachelor of Engineering in the field of Mechanical Engineering in 2005. After a two year stint with The Royal Bank of Scotland and Infosys he worked towards his Master of Science degree in Information Security at Northeastern University, Boston from 2007 to 2009. He is currently working as a Security Scientist at H2O.ai in California. His research interests are modeling anomalous behaviour using massive datasets and computer networking vulnerabilities. The area of his research interest includes malicious behaviour in computer networks, and financial fraud.