A SYSTEM FOR CHARACTERISING INTERNET BACKGROUND RADIATION

Submitted in partial fulfilment
of the requirements of the degree of

BACHELOR OF SCIENCE (HONOURS)

of Rhodes University

David Yates

Grahamstown, South Africa
October 31, 2014
Abstract

Internet traffic sent to addresses where no device is set up to receive it is termed internet background radiation (IBR), and has been collected and studied by numerous parties since the early 2000s. This data has been shown to provide valuable insights into malicious activity on networks, and as all IBR is by nature unsolicited, there is no need to filter out legitimate traffic from the datasets before using it performing botnet and worm-related analysis.

The primary aim of this project was to develop a set of tools for exploring and characterising this data, on an historic basis. The datasets used for this purpose were collected from five network telescopes operating at Rhodes University from late 2013 to early 2014. A secondary aim is to use this set of tools to perform analysis on the data and discover significant trends that can then feed into further development of the system.
ACM Computing Classification System Classification

Thesis classification under the ACM Computing Classification System (2012 version, valid through 2014) [16]:

D.7.7 [Network services]: Network monitoring

D.7.6 [Network services]: Network management

General-Terms: Metrics, Experimentation
Acknowledgements

First and foremost, immense gratitude is due to my supervisor, Barry Irwin, who provided crucial guidance and resources for the completion of this project.

I’d also like to thank Adam Schoeman, who recommended the use of Elasticsearch and Kibana to index and query the packet data. Without these technologies, the functionality and efficiency of the code produced in this project would be significantly poorer.

Thanks to my parents for supporting me during my studies and understanding when I was unable to visit because I was busy working on this project. And to my cat Marmite, who missed me a lot while I was away studying instead of scratching his chin.

The 2014 Rhodes Computer Science Honours class also more than deserve my thanks for being a great bunch to share a lab with and an immeasurable enrichment to my Honours year.

Thanks also to the open source community and the developers of the many useful libraries and tools used to produce this research.

This work was undertaken in the Distributed Multimedia CoE at Rhodes University, with financial support from Telkom SA, Tellabs, Genband, Easttel, Bright Ideas 39, THRIP and NRF SA (TP13070820716). The authors acknowledge that opinions, findings and conclusions or recommendations expressed here are those of the author(s) and that none of the above mentioned sponsors accept liability whatsoever in this regard.
Contents

1 Introduction 1
   1.1 Problem Statement .............................................. 1
   1.2 Research Goals ................................................ 2
   1.3 Research Scope ................................................ 2
   1.4 Document Conventions ......................................... 3
   1.5 Research Approach and Document Structure .................... 3

2 Literature Review 5
   2.1 Internet Background Radiation .............................. 6
   2.2 Components of Internet Background Radiation .............. 8
      2.2.1 Worms .................................................. 8
      2.2.2 Scanning Activities ..................................... 9
      2.2.3 DDoS attack backscatter ................................ 10
      2.2.4 Misconfigurations ...................................... 12
   2.3 Border Gateway Protocol ..................................... 12
   2.4 Network Traffic Collection, Characterisation and Classification 14
      2.4.1 Collection of IBR data ................................. 14
## Contents

2.4.2 Network Telescopes (Darknets) ........................................... 14  
2.4.3 Greynets ................................................................. 15  
2.4.4 Analysis of IBR data ..................................................... 16  
2.4.5 Packet-level analysis .................................................... 16  
2.4.6 Network Flows .......................................................... 17  
2.4.7 Honeynets ............................................................... 17  
2.5 Summary ................................................................. 18  

3 Data, Tools and System Development ........................................... 19  
3.1 Datasets ........................................................................... 19  
3.1.1 Data collection ............................................................. 20  
3.1.2 146 and 155 (Category A) ............................................... 20  
3.1.3 196-A, 196-B and 196-C (Category B) ............................. 21  
3.2 Tools Used ........................................................................ 21  
3.2.1 Python ......................................................................... 22  
3.2.2 Elasticsearch ................................................................. 22  
3.2.3 Kibana ......................................................................... 23  
3.2.4 Additional Libraries ....................................................... 25  
3.2.4.1 Scapy ..................................................................... 25  
3.2.4.2 Pandas .................................................................... 25  
3.2.4.3 Matplotlib ................................................................. 25  
3.2.4.4 OpenCV ................................................................. 26  
3.2.4.5 Elasticsearch Python API .......................................... 26
CONTENTS

3.3 Scripts Developed ................................................. 26
  3.3.1 PCAP File Loader Script .................................... 26
  3.3.2 Elasticsearch Interface Script ................................. 27
  3.3.3 Graphing Script ................................................ 27
  3.3.4 Scanning Graph Script ......................................... 28
  3.3.5 Spectrum Script ................................................ 28
3.4 Workflow ............................................................. 29
3.5 Timing and Storage Metrics ......................................... 31
  3.5.1 Timing ............................................................ 31
    3.5.1.1 Data Loading, Indexing and Re-indexing ................. 32
    3.5.1.2 Data Exploration .......................................... 32
    3.5.1.3 Data Analysis ............................................. 33
    3.5.1.4 Timing Conclusion and Recommendations ............... 34
  3.5.2 Storage .......................................................... 34
3.6 Summary ............................................................ 34

4 Analysis of Results .................................................. 36
  4.1 Analysis Approach: Case Studies ................................. 36
  4.2 Case Study of TCP/22 .......................................... 38
  4.3 Case Study of TCP/3389 ........................................ 43
  4.4 Case Study of UDP/1434 ....................................... 44
  4.5 Summary .......................................................... 48
List of Figures

2.1 Packets per hour by type over one week in an unused /8 (Taken from Pang et al., 2004) .................................................. 7

2.2 Diagram demonstrating DDoS backscatter ........................................ 11

2.3 BGP routing between Autonomous Systems ..................................... 12

2.4 Diagram of a greynet with two listeners deployed on a small sample network 15

3.1 Screen capture of Kibana running the system’s custom-configured Darknet Dashboard .................................................... 24

3.2 Example colour spectrum .............................................................. 28

3.3 Example of the destination IP address colour spectrum script’s output for packets sent between a single source address and multiple destinations within the 196-A darknet IP space ........................................ 30

3.4 Diagram of the system usage workflow ........................................... 31

4.1 TCP/22 for all five sensors ............................................................ 40

4.2 Scanning graphs of TCP/22 traffic from a single source IP (22-1) both overall and for six individual destination addresses (22-A–F) ................. 41

4.3 Colour spectrum of the earliest captured 1000 TCP/22 packets sent from source address A to addresses within the 146 darknet ......................... 42

4.4 TCP/3389 for all five darknets ....................................................... 43
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>Scanning graphs of TCP/3389 traffic from a single source IP (3389-1) both overall and for six individual destination addresses (3389-A–F)</td>
<td>45</td>
</tr>
<tr>
<td>4.6</td>
<td>Colour spectrum of the earliest captured 1000 TCP/3389 packets sent from source address A to addresses within the 155 darknet</td>
<td>46</td>
</tr>
<tr>
<td>4.7</td>
<td>UDP/1434 for all five darknets</td>
<td>47</td>
</tr>
<tr>
<td>4.8</td>
<td>TCP/1433 for all five darknets</td>
<td>47</td>
</tr>
<tr>
<td>4.9</td>
<td>Scanning graphs of UDP/1434 traffic from a single source IP (1434-1) both overall and for six individual destination addresses (1434-A–F)</td>
<td>49</td>
</tr>
<tr>
<td>4.10</td>
<td>Colour spectrum of hosts within the 196-B darknet scanned by 1434-1</td>
<td>50</td>
</tr>
</tbody>
</table>
List of Tables

3.1 Total packet counts for the 146 and 155 datasets . . . . . . . . . . . . . . . . . . . . . 20
3.2 Traffic composition percentage for the 146 and 155 datasets . . . . . . . . . . . . . . 20
3.3 Total packet counts for the 196/8 datasets . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
3.4 Packet types by percentage for the 196/8 darknets . . . . . . . . . . . . . . . . . . . . . 21
3.5 Table of the graphs shown in the Kibana Darknet Dashboard . . . . . . . . . . . . . 24
3.6 Table of system specifications for the two computers used by the system . . . . 32
3.7 Table of timing data for each of the system’s analysis scripts . . . . . . . . . . . . . 33
4.1 Top 10 destination ports for each dataset, UDP and TCP traffic combined
(ports which receive a majority of UDP traffic shaded) . . . . . . . . . . . . . . . . . . . 37
4.2 Top 10 destination ports for each dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
4.3 Top 10 destination ports for each dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
### List of Code Listings

1. **ESInterface** filter dictionary example ........................................... 27
2. **ESInterface** filter dictionary example ........................................... 27
3. Excerpt from the SSH authentication logs on a server on the Internet, showing a brute-force attack ................................................................. 39
4. Excerpt from the SSH authentication logs on a server on the Internet, showing a DNS spoofing attack ................................................................. 39
5. JSON-encoded structure of the Elasticsearch “packets” index used to store the data .......................................................... 61
6. Alternate JSON-encoded structure of the Elasticsearch “packets” index used to store the data ............................................................. 62
7. loadpcap.py ....................................................................................... 63
8. esinterface.py .................................................................................. 64
9. graph.py ......................................................................................... 65
10. spectrum.py ..................................................................................... 65
11. scanning.py ................................................................................... 66
Chapter 1

Introduction

The aim of this research is to investigate and characterise a subset of packet data active on the Internet known as internet background radiation (IBR). This traffic is largely illegitimate and often contains reflected traffic and evidence of programmatic scans that can provide many insights into malicious activities online.

Every year, millions of packets of data are sent to targets not set up to receive them. Although some sources of this data are simply misconfigured network adapters, other causes include malicious activity such as distributed denial-of-service (DDOS) attacks, computer worms and botnets (Pang et al., 2004; Wustrow et al., 2010). A computer worm is a kind of virus that, once executed, duplicates itself and spreads to other computers over a network (Staniford et al., 2002; Zou et al., 2005). The most well-programmed worms are able to propagate through a network within minutes and can do so indefinitely. A DDOS attack occurs when numerous systems (usually compromised) all attack a single system, flooding it with packets to the point where it is too overloaded to serve ordinary users (Moore et al., 2006). Both of these phenomenons are harmful to users and content providers on the internet, and both can happen very suddenly, leaving victimised systems with little time to react.

1.1 Problem Statement

IBR is the phenomenon of useless and largely illegitimate network traffic on the Internet. It can be imagined as analogous to a multitude of physical letters sent to physical addresses that do not exist, such as non-existent street numbers on real streets. Despite ostensibly
1.2. RESEARCH GOALS

being a waste of bandwidth, this traffic can be useful to network researchers interested in the spread of worms and the occurrence of DDoS attacks. Analysing this traffic can allow researchers to make inferences about the general state of malicious activity on the Internet.

This project aims to create a system for characterising historical IBR captures in order to further the study of this phenomenon, and as a step towards the development of near-realtime, intelligent systems that perform the same function.

1.2 Research Goals

The main goals of the research are as follows:

1. To develop a proof of concept system for conducting analysis on IBR data, focusing on speed, extendability and flexibility.
2. To demonstrate the use of this system on sample data sourced from Rhodes University’s five network telescopes.
3. To identify trends in the data that can provide the basis for further extensions to the system and future work in the research area.

1.3 Research Scope

This investigation used IPv4 packet data from the Rhodes University network telescopes (Irwin, 2011, 2013a). Factors analysed are packet source and destination ports, source and destination addresses and protocols, all analysed with respect to temporal changes (including diurnal and other cycles and the general progression of time).

This project aims to contribute towards the eventual development of a near real-time monitoring system to detect port scans, worm activity and DDoS attacks, but developing that kind of system is outside the scope of this particular project. Analysis work will be focused on historical data. Another aim this project will contribute towards but not directly attempt is the automatic generation of similar data based on the characterisations achieved.
Background information on Border Gateway Protocol (BGP) routing is provided in Chapter 2, towards the investigation of the use of this and related systems in determining internet protocol (IP) address spoofing. This was originally considered as a potential element of this project at its inception, but later discarded as out of scope.

1.4 Document Conventions

In this document, the CIDR notations /8, /16 and /24 will be used to refer to the traditional pre-CIDR Class A, Class B and Class C subnets (Fuller et al., 1993) respectively, in order to maintain consistency with the referencing of differently sized IPv4 subnets (such as /19s). The number following the “/” denotes the number of fixed bits (out of 32 total bits) in each of the addresses within the subnet.

The protocol/port notation shall be used to denote packet types according to their transport layer protocols – Transport Control Protocol (TCP) User Datagram Protocol (UDP) or Internet Control Message Protocol (ICMP) – and destination ports (excepting the case of ICMP packets, which have no associated ports). For example, TCP/22 denotes TCP packets on port 22, and UDP/1433 denotes UDP packets on port 1433.

In places were third-party tools and libraries have been mentioned, mainly in Chapter 3, web links to the official websites of the tool or library mentioned are provided as footnotes. In the absence of an official website, direct links to code repositories are given.

1.5 Research Approach and Document Structure

The research task was approached in the following stages, which also indicate the structure of the remainder of this document:

1. First, a literature review was conducted to survey and summarise the current state of research in the field of IBR and network traffic analysis. This forms Chapter 2 of the document.

2. Following the literature review, the IBR data from the aforementioned Rhodes University network telescopes was sourced and the titular system for characterising IBR was designed and constructed, with the aid of a number of external tools and libraries. This process is detailed in Chapter 3.
3. The system was then used to analyse the datasets sourced, as a demonstration of its functionality and the kind of analysis document it could be used as an aid in constructing. This is shown in chapter 4, which contains an analysis of some key points of the data, concentrating on three case studies of different destination ports.

4. Finally, the research was concluded. Chapter 5 gives the conclusions reached, evaluates the results achieved with respect to the goals set out in this chapter, and suggest topics and directions for future related work.
Chapter 2

Literature Review

This chapter introduces the reader to Internet Background Radiation (IBR), both its components and common recording practices. In doing so, it will set up a theoretical basis for the construction of the IBR characterisation system to follow.

The literature review is split into three main parts. Section 2.1 provides background information on IBR and its components: worms, scanning activities, backscatter from reflected distributed denial-of-service (DDoS) attacks, and misconfigurations. These are the sources of the data being characterised. Section 2.3 discusses the Border Gateway Protocol (BGP), the protocol used on the Internet for routing packets between Autonomous Systems (ASs). This is of interest as the construction of live BGP tables for the collected data may provide a way of discovering spoofed IP addresses. Section 2.4 provides information on IBR collection tools and analysis methods. The collection tools discussed are darknets (also called network telescopes, sinkholes, blackhole monitors or background radiation monitors (Bailey et al., 2006)) and greynets (Harrop & Armitage, 2005). In this literature review, darknets and greynets will both referred to as network telescopes, with the terms “darknet” or “greynet” being used when more specificity is required.

The analysis tools discussed are packet-level analysis, network flows and and honeynets. This section also delves into previous work in characterising network activities with statistical tools. This will form a basis for the system implemented in this study.
2.1 Internet Background Radiation

The term internet background radiation (IBR) was coined by Pang et al. (2004) to describe an ongoing variety of unproductive network traffic destined for addresses not set up to receive it. Traffic of this nature results from four main sources: worm and virus activity, network reconnaissance scans, backscatter from distributed denial of service (DDoS) attacks and misconfigured networking equipment such as routers and servers (Pang et al., 2004). Pang et al. (2004) discovered that packets with the TCP SYN-ACK and TCP RST flags set made up the majority of the darknet data recorded over four days on a /19 network, over one week on ten adjacent /24 networks and over one week on a /8 network. This dominance can be seen in Figure 2.1, taken from Pang et al. (2004), which shows the data for the /8 network used.

Apart from that, the data recorded on each network had few similarities. The extreme volatility of the IBR in comparison to ordinary, productive traffic (that is, traffic made up by legitimate connections without malicious intent or misconfiguration on either side) was noted, as was the potential difficulty of discovering new types of traffic, especially new types of worms, as they may be intentionally designed to use the same ports as other worms (Pang et al., 2004).

Important advances in detecting and characterising IBR include determination of packet sources (as packet source addresses can be forged – this is called spoofing) (Barford et al., 2006) and various advances in darknet placement and configuration and the analysis of data collected by darknets (Bailey et al., 2005, 2006).

Beginning in October 2008 and continuing through 2009 and 2010, TCP packets destined for port 445 – products of the Conficker worm – became a large contributor to IBR, significantly increasing its prevalence (Wustrow et al., 2010). This is still true today (Irwin, 2013a).

Wustrow et al. (2010) discovered that modern IBR is increasingly made up of TCP SYN packets, with the total percentage of TCP SYN packets increasing from 62.7% of the total in 2006 to 93.9% in 2010, and decreasingly of the TCP SYN-ACK packets that formed the majority of the data recorded by Pang et al. (2004), and which accounted for 26.1% of the total packets in 2006 but only 5.2% in 2010. Conficker’s prevalence has also homogenised disparate IBR datasets to a much greater degree than discovered by Pang et al. (2004) (Wustrow et al., 2010; Irwin, 2013b).
2.1. INTERNET BACKGROUND RADIATION

The homogenisation of IBR data in the time since Pang et al. (2004) is further corroborated by Irwin (2013a), who discovered a significant degree of similarity across five /24 blocks occupying far apart areas in IPv4 address space, even extending to packets not characterised as products of Conficker. Nkhumeleni (2014) conducted further research into correlations between the datasets of these five /24 blocks, noting that ICMP packet traffic in the datasets showed similarities regardless of their network block locations. After removing the Conficker data (all packets destined to TCP port 445) Nkhumeleni (2014) found a strong cross-correlation between the African network blocks surveyed and a moderate cross-correlation between the North American network blocks surveyed.

Of some interest for the future is the advent of IPv6 IBR. At the moment it accounts for a very small percentage of both total IBR and total IPv6 traffic and has been characterised as largely caused by equipment misconfiguration, but this may change as IPv6 sees more wide-scale adoption (Czyz et al., 2013).

Figure 2.1: Packets per hour by type over one week in an unused /8 (Taken from Pang et al., 2004)
2.2 Components of Internet Background Radiation

The four main components of IBR can be broadly categorised into two groups: active and passive traffic. Active traffic encompasses worms and scanning activities, both of which seek a response from the address they are destined for and are often malicious. Passive traffic encompasses DDoS backscatter and misconfigurations, which are merely the end results of DDoS attacks and network device misconfiguration, respectively, and carry no expectation of responses from other devices on the network (Irwin, 2011).

2.2.1 Worms

A worm is a form of virus that is network-aware and self-propagates across a network, using infected devices as springboards from which to infect other devices, often at random (Staniford et al., 2002; Zou et al., 2005). As a result of this random and rapid propagation, worms contribute significantly to IBR by scanning random and potentially unused addresses in search of vulnerable systems. The worm scanning activity that contributes to IBR is largely on the TCP protocol, but a notable exception to this is the SQL Slammer worm, which scans the UDP 1434 port (Wustrow et al., 2010).

Staniford et al. (2002) profiled the Code Red I, Code Red II and Nimda worms. Code Red I and II both self-propagated by spreading to randomly generated IP addresses, although notably a bug in Code Red I’s random generation meant that the same seed was used for all random generation and thus the worm was not able to spread as far as it had presumably been designed to.

Code Red II, a worm that exploited the same Microsoft IIS web server vulnerability (CVE-2001-0500) as Code Red I but was otherwise unrelated, did not share this defect and was able to propagate far more successfully. Code Red II selected IPs to which it attempt to spread in the following manner: four out of eight times, it would choose an address from its host’s /8 address space, three out of eight times, it would choose an address from its host’s /16 address space, and the remaining one out of eight times, it would choose an address from the entire IPv4 address space. This allowed the worm to spread quickly across internal networks, taking advantage of the likelihood that hosts with close IPs tend to be close together within the network.

\[1\text{http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001-0500}\]
Staniford et al. (2002) noted the difficulty of analysing the behaviour of Code Red II given that it was active at the same time as Code Red I, and as both made use of the same vulnerability and had generally similar behaviour, it was difficult to assign specific traces of Code Red behaviour to one or the other, especially if packet payloads were not taken into consideration. This is because the header data of packets from Code Red I and Code Red II would both have the same source and destination ports and the same protocols, with the only obvious difference being that Code Red II packets targeted a wider range of addresses than Code Red I packets.

The first Internet worm monitoring system was introduced by Zou et al. (2005). The system detected worm activity by identifying trends in network traffic and relating them to models of worm propagation.

The advent of the Conficker worm is primarily responsible for the significant increase in the percentage of IBR in Internet traffic (Wustrow et al., 2010). Conficker exploited a vulnerability in the Windows Server service that allowed for arbitrary code execution on receipt of a specially formed RPC message. The worm has infected between 7 and 15 million hosts and is still spreading (Shin & Gu, 2010).

Irwin (2012) conducted an analysis of the Conficker worm’s evolution based on IBR packets collected during the worm’s 2008 outbreak. The packets identified as resulting from Conficker’s activities were mostly TCP packets destined for port 445 (used for Microsoft Active Directory and Windows shares) sent from hosts running a Microsoft Windows family operating system and targeting certain ranges, which matched the nature of the worm’s operations.

### 2.2.2 Scanning Activities

Port scanning is the process of discovering information about a network by probing its devices with packets. It is often carried out by attackers or worms. By discovering which ports are open, closed and filtered on various devices on a network, attackers can discover IP addresses and their associated MAC addresses on a network, as well as what services are running and what firewall rules are in place, giving them enough information to launch an attack optimised to take advantage of any existing vulnerabilities (Modi et al., 2013).

---

2https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt
A common goal for attackers is the creation of a botnet (Staniford et al., 2002). A botnet is the collective term for all computers compromised and controllable by a single controllable entity over a network connection. Botnets are closely related to both worms and DDoS attacks, in that a common means of generating and extending a botnet is writing a worm that infects hosts with bot software, and a common use for botnets is performing large-scale DDoS attacks (Cooke et al., 2005). Botnets also give their controllers access to sensitive data on victimised systems and allow for impersonation of users (Staniford et al., 2002).

Wustrow et al. (2010) noted an increase in SSH scanning activity to the point of significantly contributing to IBR, starting in 2007. Concurrent with this was the rise of TCP port 23 (traditionally used for Telnet connections\(^3\)) scanning, indicating increased attempts to discover back doors installed by worms.

Work has been done on systems like BotHunter (Gu et al., 2007) and BotFinder (Tegeler et al., 2012) to discover and identify the malware family (a system of categorisation of malware types by shared behavioural patterns (Rieck et al., 2008)) of bots by the network traffic they produce. BotFinder was able to identify botnet-infected systems and botnet activity without analysing the content of any of the packets in the inspected network traffic (Tegeler et al., 2012).

Bou-Harb et al. (2013) proposed a system of detecting network scans with a focus on the targets of the scans rather than their sources. The claim made was that as it is sometimes not possible to determine the sources of scans, techniques relying on this determination were prone to being less effective. The Bou-Harb et al. (2013) system instead chooses to cluster similar scans together under the assumption they come from the same source as an alternative to finding their sources.

### 2.2.3 DDoS attack backscatter

In a DDoS attack, one or more clients send a large amount of non-productive packets to a host, with the intent of either crashing or slowing the software running on that host or overwhelming the host’s physical resources such as processing power and memory. The packets sent in this kind of attack are often spoofed: that is, their source addresses have been fabricated and do not match the addresses of the systems they were sent from (Moore et al., 2006).

\(^3\)http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
2.2. COMPONENTS OF INTERNET BACKGROUND RADIATION

A TCP SYN packet created in this way will prompt a response to its source address in the form of a TCP SYN-ACK packet, as part of the three-way handshake (Cerf et al., 1974). When a victimised host sends responses such as these to packets with spoofed IP addresses, those responses will go to the spoofed IP, which may be an address residing in unused address space. This is especially probable if the spoofing is done randomly rather than with the intention of implicating a third party in the attack (Moore et al., 2006). These response packets are termed “DDoS backscatter”. Figure 2.2 demonstrates this.

DDoS backscatter makes up a segment of IBR recorded by darknets and can be used to analyse DDoS activity. This technique was first used by Moore et al. (2006). The use of backscatter analysis in studying DDoS activity has been criticised by Mao et al. (2006), who cites a lack of address spoofing in most DDoS attacks conducted.

Fachkha et al. (2014) discusses a way of detecting DNS amplification–based DDoS activity from darknet data without relying on backscatter packets. This is discussed further in Section 2.4.6. The study highlights potential active components of IBR that are related to DDoS attacks but do not form part of passive backscatter.
2.2.4 Misconfigurations

Occasionally a network device on a host or a router is accidentally set up to route packets to addresses within unused space monitored by a darknet. This creates small, benign set of IBR data known as misconfigurations (Pang et al., 2004). Misconfigurations account for the majority of IPv6 IBR at present (Czyz et al., 2013). The most common misconfigurations are unused addresses mistakenly entered in the address field, either in the systems themselves or in the Network Address Translation (NAT) routers (Irwin, 2011).

2.3 Border Gateway Protocol

A possible way of determining whether source addresses of packets in IBR are spoofed is to maintain a set of BGP routing tables for the data (Yao et al., 2010). BGP routing is the system that governs how autonomous systems (ASes) route traffic to each other. This is shown in Figure 2.3. An AS, also called a routing domain, is a network with a clearly defined routing protocol between hosts. A single AS is often owned by an ISP.4

BGP is an application layer protocol overlaying TCP. BGP messages (sent over TCP port 179) generally specify a network, a subnet and a number of attributes, the most

---

4http://searchnetworking.techtarget.com/definition/autonomous-system
relevant of which is the AS-path attribute, which indicates the order in which a packet must be transferred from AS to AS in order to reach the specified subnet of the specified network. Routers receive BGP route advertisement messages, prepend their own AS numbers to them, and then send those messages out as route advertisements to other routers (Schluting, 2006).

Generally, implementations of BGP routing include mechanisms for filtering (Schluting, 2006) and route flap damping (Villamizar et al., 1998). Filtering allows routers to reject route advertisements from unexpected sources. Route flap damping causes routers to ignore a route that disappears and reappears suddenly two or more times in quick succession (called flapping), with the time period in which to ignore the offending prefix increasing exponentially every time the route flaps.

Traditionally, BGP routing has few internal security features. All announcements of routes by all ASes are regarded as true. This raises security concerns, as BGP routes are trivially spoofed in the current system. Research has been done to elucidate the weaknesses of BGP routing and to attempt to make it more secure.

One of the earliest attempts at securing BGP was the Secure Border Gateway Protocol (S-BGP) (Kent et al., 2000), a set of cryptographic attestations by which ASes could broadcast their veracity. More analytically, Qiu et al. (2007) discovered that legitimate BGP routes had stable historical structures and thus spoofing could be identified by comparing the current state of a routing system with its historical state.

Song et al. (2013) showed that S-BGP and similar systems do not successfully secure BGP against spoofing, as attackers can spoof routes indirectly through fundamental vectors S-BGP was not designed to protect against. The highly configurable nature of BGP implementations means that each AS can have different policies for route flap damping, filtering and the minimum route advertisement interval (MRAI) and thus an entire route can be invalidated by containing just one AS with badly configured policies (either due to ignorance or malicious intent).

Although it is widely known that BGP routing is vulnerable to attacks and many secure variations have been proposed, none have seen wide-scale adoption (Chan et al., 2006).
2.4 Network Traffic Collection, Characterisation and Classification

The study and collection of IBR is a richly researched area that can be broadly split into two parts: the creation and refinement of tools for collecting IBR data, and the analysis of and extraction of meaning from the IBR data collected.

2.4.1 Collection of IBR data

IBR traffic is recorded via network telescopes, which are systems that monitor areas of unused address space and record packets sent to them (Moore et al., 2004). Before Pang et al. (2004), network telescopes were used to study DDoS backscatter (Moore et al., 2006) and worm activity (Moore et al., 2003a), but no overarching characterisation was done on the data.

Network telescopes variants are called darknets, greynets and other names such as dimnets (Irwin, 2011) based on the amount of unused addresses within or “darkness” of the address space they are set up to monitor.

2.4.2 Network Telescopes (Darknets)

A darknet is a system that monitors traffic towards unused address space within a network (Moore et al., 2004). A darknet is hosted on a network address that does not send any packets or interact with the outside network. Darknets set up to monitor small, well-defined address spaces will send address resolution protocol (ARP) replies to routing requests for unused addresses in the network space. When deploying darknets to monitor larger address spaces (thousands of addresses and above), a router will route the entire unused address block to the darknet instead (Bailey et al., 2006).

Legitimate traffic will not be captured by a darknet, as it has no reason for accessing addresses not set up to receive any traffic, and so all darknet traffic can be presumed to be unproductive and part of IBR (Bailey et al., 2006). Thus, there is no need to identify and separate out legitimate traffic from illegitimate and possibly malevolent traffic when analysing IBR.
2.4. NETWORK TRAFFIC COLLECTION, CHARACTERISATION AND CLASSIFICATION

Care needs to be taken to ensure that darknets are correctly positioned and configured so that they can record datasets of optimal usefulness. For example, darknets that monitor an area close to live address space received significantly more packets than ones further away (Bailey et al., 2006).

### 2.4.3 Greynets

A greynet is a variant of a network telescope that, instead of monitoring a large block of contiguous unused address space like a darknet, monitors unused addresses scattered amongst a populated area of address space, within an organisational network for example. A greynet can thus be deployed on an enterprise network for the purpose of more direct monitoring than is possible with a darknet (Harrop & Armitage, 2005).

Thusly, a greynet can be deployed in a similar manner to the deployment of a darknet over a small address space as discussed above. A greynet can have many “listeners” – systems that receive unsolicited requests – spread out over the network. This is shown in Figure 2.4.

The only difference is that a greynet’s listeners will send ARP replies to significantly less than one hundred percent of the routing requests to the address space.

Figure 2.4: Diagram of a greynet with two listeners deployed on a small sample network.
2.4. NETWORK TRAFFIC COLLECTION, CHARACTERISATION AND CLASSIFICATION

2.4.4 Analysis of IBR data

The use of statistical analysis to characterise packet data in a network is a well-explored area. Paxson (1994) constructed analytical models of network flows created by applications using data such as packet length and flow duration.

Lin et al. (2009) expanded on the work done by Paxson (1994) by included application protocol-level data in the analysis conducted. Lin et al. (2009) observes that packet size distribution (PSD) varies greatly between applications but is generally similar within applications. This provides a vector by which to cluster packet data according to application.

Barford et al. (2002) used signal analysis to identify anomalies in general network traffic collected on a border router at the University of Wisconsin and classify them into “long lived” (mass downloading of popular new software) and “short lived” (network outages, attacks and measurement anomalies) events.

2.4.5 Packet-level analysis

The most obvious way of analysing IBR datasets is to look at the trends in packet composition over time. Packet source address, destination address, protocol (usually TCP or UDP), source port and destination port are among the most commonly graphed data. Much can be determined about activity occurring on the network just by identifying trends in these data – how many unique source and destination addresses exist, the number of similar packets sent to each port and address, diurnal changes in packet behaviour and so forth.

For example, Conficker activity is evidenced by a large number of TCP SYN packets with destination port 445, occurring within the 1/4 of Internet address space Conficker propagation is limited to as a result of a bug in its psuedo-random number generation (Wustrow et al., 2010). Another indicator of Conficker activity is that whereas most legitimate TCP connections are initiated by host sending three SYN packets (in case of data loss), Conficker only sends one or two (Aben, 2009).

DDoS activity can be inferred from large volumes of TCP SYN-ACK packets, with each single source being DDoS victim responding to TCP SYN packets with spoofed IP addresses, as discussed in Section 2.2.3.
2.4. NETWORK TRAFFIC COLLECTION, CHARACTERISATION AND CLASSIFICATION

2.4.6 Network Flows

A network (or “message”) flow is a set of packets sent between a given source address and a given destination address and port (Kerr & Bruins, 2001). Network flow identification is used by routers to determine correct port numbers for routing packets and to control access to them. Network flows can also be used for traffic analysis, providing a level of detail between general Simple Network Protocol (SNMP) statistics and highly detailed packet-level data (Sommer & Feldmann, 2002).

Network flows have been used to analyse traffic for cyber-defence purposes, i.e., identifying scanning, worm and DDoS attack activity (Chickowski, 2013; Yurcik, 2005). An example of this is Fachkha et al. (2014), who proposed a method of inferring DNS amplification-based DDoS attack activity by analysing DNS netflows discovered in darknet data.

DNS queries in the darknet space surveyed by Fachkha et al. (2014) were classifiable in three categories: DNS queries with spoofed source addresses sent by an attacker (the spoofed addresses being the victim’s address), DNS queries sent by a compromised victim controlled by an attacker, and DNS queries sent as a scanning activity to infer the locations of open DNS resolvers. The sources of DDoS activity were to be inferred by the source addresses of the third type of DNS flow, wherein the source addresses of the packets would not be spoofed as the attacker would need to receive replies to carry out scanning activity successfully.

2.4.7 Honeynets

A honeynet is a system of hosts purposefully made vulnerable and exposed to worms for the purpose of studying the worms. Honeynets are often used in conjunction with darknets in systems for detecting malware and attacks. In this kind of system, a darknet or system of darknets identifies new threats in the traffic gathered and proxy the appropriate traffic to the system of honeynets for more in-depth analysis (Bailey et al., 2005).

The principle difference between darknets and honeynets is that whereas darknets just record packet data, honeynets send appropriate responses to received packets in order to study their activities further. Darknets just record packet header data, whereas honeynets allow for study of actual packet payloads.

[^5]: [http://searchsecurity.techtarget.com/definition/honeynet](http://searchsecurity.techtarget.com/definition/honeynet)
Yegneswaran et al. (2005) developed a system for providing network security personnel with in-depth network situational awareness (summarised, accurate data about moment-to-moment happenings within the network) using a honeynet system deployed on unused address space. The system highlighted two kinds of events: the advent of activity not previously seen on the network, and atypically large spikes of previously seen activities.

2.5 Summary

The study and automated analysis of IBR can provide network administrators and network security experts with valuable intelligence on the potentially malicious scanning and DDoS activities occurring in and adjacent to their networks. It is also a useful dataset for the study of worm and botnet propagation.

As IBR is packet data sent to unused addresses, it includes no legitimate connections or service traffic and individual packets or netflows can all be assumed to result from worm activity, botnet and reconnaissance scanning activity, DDoS activity and occasionally the activity of misconfigured network devices.

BGP provides a method for routing packets between ASs, and live BGP tables can be incorporated into IBR monitoring systems to identify spoofed IP addresses.

IBR is monitored using darknets and greynets. Activities in the wider Internet can be inferred through analysing the packet data and network flows occurring within these datasets, and by sending data seen as “interesting” (novel or unusual in some manner) to honeynets for analysis of their packet payloads (Yegneswaran et al., 2005).

As more sophisticated Internet worms, botnets and DDoS attack techniques are developed, and as active worms such as Conficker continue to spread, the volume of IBR will continue to increase. By studying and characterising the data, researchers and security professionals have and can continue to develop robust warning systems and countermeasures to these threats.
Chapter 3

Data, Tools and System Development

This chapter describes the datasets analysed in this document and the tools used to build the titular system to analyse them. The data, detailed in Section 3.1, comprises five sets of packet captures all representing the same time-frame, each from a separate network telescope, all of which are hosted by Rhodes University. The tools used consist of existing solutions for general problems such as packet capture (PCAP) file processing, data storage and querying and aggregation, and graphing. These tools are enumerated in Section 3.2.

These tools are used to develop a system for analysing IBR, which comprises a set of Python scripts and to which the remainder of the chapter is dedicated. A high-level overview of the workings of these scripts is given in Section 3.3. The workflow users of the system are intended to follow is shown in Section 3.4, and Timing and Storage information about the system is provided in Section 3.5. Finally, all of the above is summarised in Section 3.6.

3.1 Datasets

Data used in this report comprises of the total packet captures over five network telescopes operated by Rhodes University over a period of approximately six months – between 7 July 2013 and 12 February 2014. Research has been done on data from these sensors in the past, such as Irwin (2011, 2013a) and Nkhumeleni (2014).
To avoid pollution of the darknet data, the specific subnets over which the five darknets used as input are deployed are not given. The sensors will be referred to throughout this report as 146, 155, 196-A, 196-B and 196-C.

A previous study on data collected from these network telescopes, Nkhumeleni (2014), split the datasets into two categories based on internal similarity of the data and relative closeness of the monitored areas of IP space. These categories, termed Category A and Category B, are preserved in this document, where 146 and 155 form Category A, and 196-A–C form Category B.

### 3.1.1 Data collection

The darknet datasets for 196 and 146 were collected by two network telescopes operating in four /24s within Rhodes University’s IP spaces in the Eastern Cape. Irwin (2011) provides further details regarding the setup and configuration of these darknets. The 155 dataset was sourced from a network telescope which operated in the Western Cape during the collection period.

### 3.1.2 146 and 155 (Category A)

The 146 network telescope was launched at Rhodes University in August 2009 and the 155 darknet in the Western Cape in early 2011 Nkhumeleni (2014).

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Packet count</th>
</tr>
</thead>
<tbody>
<tr>
<td>146</td>
<td>8663877</td>
</tr>
<tr>
<td>155</td>
<td>9256729</td>
</tr>
</tbody>
</table>

Table 3.1: Total packet counts for the 146 and 155 datasets

<table>
<thead>
<tr>
<th>Sensor</th>
<th>TCP (%)</th>
<th>UDP (%)</th>
<th>ICMP (%)</th>
<th>Other (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>146</td>
<td>68.93</td>
<td>26.04</td>
<td>5.03</td>
<td>0.01</td>
</tr>
<tr>
<td>155</td>
<td>57.84</td>
<td>37.21</td>
<td>4.94</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 3.2: Traffic composition percentage for the 146 and 155 datasets
3.1.3 196-A, 196-B and 196-C (Category B)

These datasets will be collectively termed “196” throughout this document. The first one, 196-A, came into existence in August 2005. It is located in the Eastern Cape and was the first network telescope put into operation at Rhodes University (Nkhumeleni, 2014). 196-B and 196-C were launched in early 2011, along with 155, and placed on the same 196/8 IP prefix as 196-A.

<table>
<thead>
<tr>
<th>Darknet</th>
<th>Packet count</th>
</tr>
</thead>
<tbody>
<tr>
<td>196-A</td>
<td>16364786</td>
</tr>
<tr>
<td>196-B</td>
<td>15523583</td>
</tr>
<tr>
<td>196-C</td>
<td>16398029</td>
</tr>
</tbody>
</table>

Table 3.3: Total packet counts for the 196/8 datasets

<table>
<thead>
<tr>
<th>Darknet</th>
<th>TCP (%)</th>
<th>UDP (%)</th>
<th>ICMP (%)</th>
<th>Other (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>196-A</td>
<td>79.72</td>
<td>17.39</td>
<td>2.88</td>
<td>0.01</td>
</tr>
<tr>
<td>196-B</td>
<td>77.16</td>
<td>19.83</td>
<td>3.00</td>
<td>0.01</td>
</tr>
<tr>
<td>196-C</td>
<td>79.30</td>
<td>17.64</td>
<td>3.05</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 3.4: Packet types by percentage for the 196/8 darknets

3.2 Tools Used

The tools used in the analysis of the data and construction of the system are given below. The approach taken to constructing the system was to use a set of well-documented open-source tools and libraries to handle the complex, general problems such as parsing PCAP files, storing and querying data, and drawing graphs. The system constructed represents a merging of these various technologies through the mediation of a set of scripts.

The tools used consist of the Python programming language¹, detailed in Section 3.2.1, the data search engine Elasticsearch², explained in Section 3.2.2, and Kibana³, a web-based interface for Elasticsearch which is described in Section 3.2.3. A number of third-party Python libraries are also used, and these are given in Section 3.2.4.

¹http://www.python.org/
²http://www.elasticsearch.org/
³http://www.elasticsearch.org/overview/kibana/
3.2. TOOLS USED

3.2.1 Python

Python (version 2.7) was chosen as the language for the system for the following reasons:

- The language’s clear and expressive syntax allowed the researcher to focus entirely on solving the problems at hand with little concern for “boilerplate” code or otherwise dealing with difficulties in the programming itself.

- Python is greatly familiar to the researcher, and thus provides a method for turning thoughts into executable code with very little overhead.

- There exist a large selection of project-relevant libraries available, such as scapy and dpkt for processing packet captures and pandas for data processing.

- Javascript Object Notation (JSON) – the data-interchange format primarily used query Elasticsearch – and Python’s built-in dictionary data structure are largely syntactically equivalent.

Version 2.7 was chosen in lieu of newer 3.x versions of the language due to the wider external library availability in the former, including the project-crucial scapy, which was only available as a Python 2.7 library\(^4\).

3.2.2 Elasticsearch

The NoSQL document search solution Elasticsearch was used for storing and querying the data, on recommendation of a trusted third party. Other avenues such as JSON files\(^5\), Comma-separated values (CSV) files, and relational databases (PostgreSQL\(^6\) in particular) were also considered for storing and querying the extracted packet data, but ultimately Elasticsearch was chosen because of its speed and scalability (Terrier, 2013; Gormley & Tong, 2014).

It should be noted that Elasticsearch is resource-intensive, especially concerning RAM. Speed remains high for querying, but without enough RAM many queries will fail to complete. The researcher was required to reserve 6GB of RAM for Elasticsearch to ensure it would return results such as histograms of the entire (approximately) 66 million packet

\(^4\)http://www.secdev.org/projects/scapy/
\(^5\)http://json.org/
\(^6\)http://www.postgresql.org/
dataset without running out of memory. Ideally, Elasticsearch and Kibana should be run on a dedicated server for this kind of analysis, however, the setup used here was acceptable for a proof of concept.

Elasticsearch stores data in *indices*, which are conceptually similar to *databases* in traditional relational database systems. These indexes can hold multiple *types*, which are similar to tables in a relational database. These types each contain fields, each with an associated data type, much like fields in a relational database table. The structure of the Elasticsearch index, called a *mapping*, used is provided in Appendix A, Listing 5. It is given in a JSON format that can be sent directly to Elasticsearch’s index creation API in order to create the index.

A possible improvement on these index structures would be to use types with distinct mappings for each different packet protocol – in the current system, while UDP, ICMP and TCP are all different types, they all use the same default mapping. The “protocol” field is redundant, and with a custom mapping ICMP packets would have no destination or source ports defined rather than each ICMP packets setting these fields to null. This improvement was not implemented due to time constraints. It can be assumed that the the benefits would be largely cosmetic as not many ICMP packets were present in the data.

A second and more useful possible improvement to the index structure is given as an alternate mapping in Appendix A, Listing 6. This alternate mapping stores IP addresses as IP types\(^7\) rather than the strings stored by the mapping used for the system. This index was not implemented due to incompatibilities with the version of Kibana used in this project which uses Elasticsearch’s facet API to draw its graphs rather than its newer aggregations API, and as a result not only cannot take advantage of the IP address range aggregations but is also unable to search IP addresses like it can search text. A potential workaround is to store two fields for each IP address, one taking a string data type and one taking an IP address data type.

### 3.2.3 Kibana

Kibana\(^8\) is a web client for Elasticsearch built in HTML and Javascript. It provided Elasticsearch users with a dashboard from which they can easily query their data and


\(^8\)http://www.elasticsearch.org/overview/kibana/
3.2. TOOLS USED

Figure 3.1: Screen capture of Kibana running the system’s custom-configured Darknet Dashboard

display different facets of it as dynamic graphs and tables. It was used as an exploratory
interface for experimenting with the data. A custom dashboard was assembled for the
darknet data and given seven graphs, detailed in Table 3.5. The source file for this darknet
is provided with the project code repository (see Appendix B).

Kibana can be run using web-server software such as Apache\(^9\) or nginx\(^10\), and can be set
up to interface with a either a remote or a local instance of Elasticsearch. The version of
Kibana used in this project is Kibana 3.

\(^9\)http://www.apache.org/
\(^10\)http://nginx.org/

<table>
<thead>
<tr>
<th>Data</th>
<th>Graph Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet types</td>
<td>Pie chart</td>
</tr>
<tr>
<td>Source darknets</td>
<td>Bar graph</td>
</tr>
<tr>
<td>Source IP addresses</td>
<td>Bar graph</td>
</tr>
<tr>
<td>Destination IP addresses</td>
<td>Bar graph</td>
</tr>
<tr>
<td>Destination ports</td>
<td>Bar graph</td>
</tr>
<tr>
<td>Source ports</td>
<td>Bar graph</td>
</tr>
<tr>
<td>Packets over time</td>
<td>Line graph</td>
</tr>
</tbody>
</table>

Table 3.5: Table of the graphs shown in the Kibana Darknet Dashboard
3.2. TOOLS USED

3.2.4 Additional Libraries

As discussed in Section 3.2.1, one of the reasons for choosing the Python language was its large array of project-relevant libraries. The ones used in the project for the purposes of packet capture processing, data analysis and data visualisation are detailed below.

3.2.4.1 Scapy

The Python packet analysis library Scapy was used to extract data from the packet capture files analysed. It was chosen because of the thoroughness of its documentation and its lack of bugs and packet error correction facilities. In retrospect, the use of this library for processing large capture files was not the optimum choice, due to the slowness and resource-intensive nature of its PCAP processing features. Alternative libraries for Python packet processing include dpkt\(^{11}\) and libpcap\(^{12}\), the former of which was briefly used at the outset of the project but abandoned due to poor documentation and inexplicable bugs encountered with PCAP processing.

The data, comprised of approximately 66 million packets, took the computers used ten days to process and index. More information on this is provided in Section 3.5.1.

3.2.4.2 Pandas

Pandas\(^{13}\) is an open source data analysis library for Python. It provides DataFrame and Series objects for powerful modelling and manipulation of datasets, similar to and based on the models used in the statistical computing–focused R language\(^{14}\) (McKinney, 2012). Packet data stored in Elasticsearch indexes was retrieved and used to build up these DataFrame and Series objects for statistical analysis and graphing.

3.2.4.3 Matplotlib

Matplotlib’s\(^{15}\) Python graphing library pyplot was used in conjunction with Pandas to construct graphs of the data for the system. All original graphs shown in this document have been produced this way.

\(^{11}\)https://github.com/kbandla/dpkt
\(^{12}\)http://www.tcpdump.org/
\(^{13}\)http://pandas.pydata.org/
\(^{14}\)http://www.r-project.org/
\(^{15}\)http://matplotlib.org/
3.3. SCRIPTS DEVELOPED

3.2.4.4 OpenCV

The graphics and image processing library OpenCV\(^{16}\) was used to create colour spectrums representing host scans across IP space, as detailed in Section 3.3.5.

3.2.4.5 Elasticsearch Python API

The Python API for Elasticsearch was used to interface with the system’s Elasticsearch instance, although in some areas of the code this is bypassed in favour of direct used of Elasticsearch’s standard RESTful APIs\(^{17}\) through HTTP requests.

3.3 Scripts Developed

The titular system takes the form of a set of Python scripts which mediate between PCAP files, Elasticsearch and Matplotlib, making it easy to load in and analyse darknet data as well as produce graphs and other useful analysis about the data.

It was decided that the system should be produced as a set of scripts that interface with existing tools because of the power and flexibility this gives to the user of the system. A user analysing darknet data is likely to have a technical background and be comfortable with using well-documented scripts which he or she can customise and integrate into existing workflows.

3.3.1 PCAP File Loader Script

This script creates a new index on the given instance of Elasticsearch and then uses scapy to parse the given PCAP files and insert them into this index. The Elasticsearch index mapping used is shown in Appendix A, Listing 5.

A truncated version of this script containing class definitions, function definitions and docstrings is provided in Appendix B, Listing 7.

\(^{16}\)http://opencv.org/
3.3. SCRIPTS DEVELOPED

**Listing 1** ESInterface filter dictionary example

```json
{
    "destination_port": "22",
    "protocol": "TCP",
    "source_darknet": "146"
}
```

This filter will narrow down the packets on which the `date_histogram` or `top_terms` query will be run to TCP/22 packets in the 146 darknet.

**Listing 2** ESInterface filter dictionary example

```json
{
    "destination_port": "22",
    "protocol": ["TCP", "UDP"],
    "source_darknet": "146"
}
```

This filter will narrow down the packets on which the `date_histogram` or `top_terms` query will be run to UDP/22 and TCP/22 packets in the 146 darknet.

### 3.3.2 Elasticsearch Interface Script

This script implements the object `ESInterface`, which provides a set of high-level methods for querying the Elasticsearch instance and index with which it is instantiated. The two pivotal methods are `date_histogram` and `top_terms`, both of which take in parameters in the Python dictionary/JSON form shown in Listing 1 and Listing 2, which demonstrates how to provide multiple alternatives for values in a field.

All of the querying methods implemented return the query results as pandas `DataFrames` with appropriate column headings. The class definitions, function definitions and docstrings in this script are provided in Appendix B, Listing 8.

### 3.3.3 Graphing Script

This script implements the object `Graph`, which contains methods for drawing a graph of given packet data (sourced from Elasticsearch through an ESInterface). When run, this script produces multi-plot graphs of traffic over time. One graph is produced for each of the ten most popular source and destination ports for both the UDP and TCP protocols, each with a plot for each source darknet.

A truncated version of this script containing class definitions, function definitions and docstrings is provided in Appendix B, Listing 9.
3.3.4 Scanning Graph Script

This script, given a destination port, a protocol (either TCP or UDP, as ICMP lacks ports) and a source darknet, produces a set of graphs using the ESInterface and Graph objects. These graphs show the packets sent from the top ten source IP addresses for that data to the top ten of each of their destination IP addresses. These graphs provide a basis for a viewer to reason about whether or not a given address is conducting a port scan.

A truncated version of this script containing class definitions, function definitions and docstrings is provided in Appendix B, Listing 11.

3.3.5 Spectrum Script

Similar to the scanning graph script, this script produces metrics for individual source addresses that send packets to multiple hosts, but whereas the graphing script produces graphs showing time and volume data, this script outputs a colour spectrum that gives a visual indication of how the order in which target addresses were targetted relates to their ordering within their /24. The images produced are made up of coloured vertical bars, with lighter colours representing target IP addresses with a fourth octet closer to 255 and darker colours representing target IP addresses with a fourth octet closer to zero.

Figure 3.2 shows an example colour spectrum of 255 pixels in width. This would be the output of the script in the event that it found a single host scanning each possible IP address in a single /24, in ascending order from x.x.x.1 to x.x.x.255.
Figure 3.3 shows an example output of the script run against the data, in which the order of scanning is significantly more erratic than the example spectrum.

The script also produces CSV files containing the data used to create the spectrums. A truncated version containing class definitions, function definitions and docstrings is provided in Appendix B, Listing 10.

\section*{3.4 Workflow}

This section provides the process a user will follow when using the system on a new set of IBR data. This process is shown as a diagram in Figure 3.4. Note that the first step, data collection, is not handled by the system, but by a network telescope.

1. Data collection: data is collected by a network telescope, written to PCAP files and stored.

2. Data loading:
   
   (a) Elasticsearch is installed and configured on the server to be used for data analysis.
   
   (b) The PCAP files collect in the previous step are passed to the PCAP File Loader Script (Section 3.3.1), which indexes them in the Elasticsearch instance.

3. Data exploration:
   
   (a) Kibana is installed, configured and deployed using a web-server application on the server running Elasticsearch.
   
   (b) The system’s Darknet Dashboard is as the Kibana data dashboard, and dynamic exploration and drill-down analysis is performed on the data.

4. Graphing and analysis:
   
   (a) The graphing, scanning graph and colour spectrum scripts are used to produce formal metrics about the data.
   
   (b) These metrics are included in a report. Chapter 4 can be considered an example report.
Figure 3.3: Example of the destination IP address colour spectrum script’s output for packets sent between a single source address and multiple destinations within the 196-A darknet IP space
3.5 Timing and Storage Metrics

Much of the development of the system was driven by the desire to explore and produce statistics on the large amount of data used more quickly and hereby enable dynamic exploration by the researchers. Timing metrics for the execution and usage of the different system elements are given in Section 3.5.1 below. Section provides a comparison between the storage space taken up by the original PCAP files and the Elasticsearch instance of these files.

3.5.1 Timing

The timing metrics for the system are split into two sections: metrics for loading and indexing the data, metrics for exploring the data, and metrics for running analysis on the data.
3.5. TIMING AND STORAGE METRICS

<table>
<thead>
<tr>
<th></th>
<th>Server</th>
<th>Workstation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor Type</td>
<td>Intel Xeon E5-5620</td>
<td>Intel Core i5-3570 CPU</td>
</tr>
<tr>
<td>Number of Cores</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>Speed (GHz)</td>
<td>2</td>
<td>3.4</td>
</tr>
<tr>
<td>RAM (GB)</td>
<td>64</td>
<td>818</td>
</tr>
</tbody>
</table>

Table 3.6: Table of system specifications for the two computers used by the system

Two computers on the same network were used to run the system. The first was the Server, used to process and stream the data from the PCAP files. This data was streamed to an Elasticsearch server running on the second computer, termed the Workstation, which also hosted the instance of Kibana and performed all the rest of the system’s operations. The specifications of these systems are given in Table 3.6.

3.5.1.1 Data Loading, Indexing and Re-indexing

As noted in Section 3.2.4.1, the system took ten days to process all of the data (approximately 66 million packets at an average of 77 packets per second) from the PCAP files and stream them from the Server to the Workstation. Ten parallel versions of the script itself (Section 3.3.1) were run on different PCAP files on the Server.

It was determined that Scapy was the largest contributor to this extreme time frame. Scapy’s poor performance at scale has been noted by Alcock *et al.* (2012), Levillain (2014) and Claveirole (2010). It is hypothesised that using dpkt or libpcap may produce faster results. A second contributor to the lengthy time-frame is the large number of packets loaded in all at once.

Re-indexing all of the data loaded in, a process of transferring everything from one data index to a new data index, which was necessary to correct a mistake made with the data types in the first version of the Elasticsearch index mapping19, took the Workstation a single day.

3.5.1.2 Data Exploration

The data exploration was performed entirely in Kibana, running on the workstation computer. This phase consisted of informal data searching and filtering. The data fields –

---

19Refer to Appendix A for more details about Elasticsearch mappings and indices.
3.5. **TIMING AND STORAGE METRICS**

<table>
<thead>
<tr>
<th></th>
<th>Multi-plot graphs</th>
<th>Scanning graphs</th>
<th>Spectrums</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time taken (minutes:seconds)</td>
<td>00:48.55</td>
<td>05:57.72</td>
<td>10:26.23</td>
</tr>
<tr>
<td>Number of graph plots drawn</td>
<td>200</td>
<td>1650</td>
<td>N/A</td>
</tr>
<tr>
<td>Elasticsearch queries made</td>
<td>204</td>
<td>1815</td>
<td>165</td>
</tr>
</tbody>
</table>

Table 3.7: Table of timing data for each of the system’s analysis scripts

source and destination ports, source and destination addresses, packet protocol and source darknet – were filtered and unfiltered and the graphs of top values per field and data time series were analysed in search of trends.

Each data transformation attempted took an average of 11 seconds to propagate to all seven graphs drawn. A video demonstration of the use of the Kibana dashboard is provided on the project website.

### 3.5.1.3 Data Analysis

The timing data given in Table 3.7 was recorded during the construction of the graphs and spectrums used for the three traffic types given case studies in Chapter 4. These three traffic types are TCP/22, TCP/3389 and UDP/1434.

During this test, the scripts performed the following work:

- **Multi-plot graphs** (Section 3.3.3): Produced 40 graphs (the top ten source and destination ports for both the UDP and TCP protocols), each with five subplots.

- **Scanning graphs** (Section 3.3.4): Produced scanning graphs for the three traffic types. For each type, five sets of graphs were drawn, one for each darknet. Each of these sets of graphs consisted of overall traffic graphs of the top ten source addresses of the traffic, and for each of these source addresses, ten individual graphs for the top ten destinations reached.

- **Spectrums** (Section 3.3.5): Drew spectrums for three types of traffic. For each of these traffic types, spectrums were drawn for the top ten source addresses. Each spectrum was drawn with one line for each relevant packet sent by the source address, up to a maximum of one thousand lines, a maximum limit decided on to ensure the spectrums would be printable.
### 3.5.1.4 Timing Conclusion and Recommendations

It should be noted that although it takes the system a relatively long time to load in data from the PCAP files, the construction of graphs and analytics from the loaded data is relatively quick. This is beneficial given that data loading is the least frequently repeated process in the system, whereas the graphing elements of the system will be run multiple times with different configurations throughout the process of analysing the data, and the very fast Kibana dashboard will be used and queried still more frequently.

The lengthy loading time could be mitigated if the user of the system were to load smaller volumes of packets into the system at regular intervals and then performing analysis on the loaded data in the interim, rather than loading millions of packets in all at once. This approach, however, would be rendered less effective in the event of a sudden and heavy spike of traffic to the network sensor.

### 3.5.2 Storage

The original five PCAP source files loaded into the system took up a total of 7.2GB in disk space. By comparison, the Elasticsearch index containing the processed and indexed data from these files was 11.2GB in size. Note that the Elasticsearch index kept only the timestamp, protocol, the ports and the IP addresses associated with each of the packets, whereas the PCAP files contained all of the packet header fields.

### 3.6 Summary

In this chapter, the darknet data and analysis tools (both used and developed) were enumerated and described. The data comprised packet capture files from five darknets maintained by Rhodes University, taken over a period of approximately seven months. The tools comprised of the NoSQL database system Elasticsearch, the Python programming language, the Python packet analysis library Scapy, the Python data manipulation library Pandas, the Python graphing library Matplotlib and graphics library OpenCV, and a set of scripts developed to mediate between these tools and produce human-parsable results for understanding the data.
The set of scripts forms the titular system for characterisation of IBR, and were detailed in Section 3.3. Truncated versions of these scripts and a link to the publically accessible source code repository for the system are provided in Appendix B.

System timing and storage information and comparisons were given in Section 3.5. The main point noted in this section was the ability of the system to create graphs of the data relatively fast even though the initial loading of the large datasets used was quite slow.
Chapter 4

Analysis of Results

This chapter contains the analysis conducted on output produced by the system described in Chapter 3, Section 3.3 when given the packet capture data detailed in Chapter 3, Section 3.1. It can be considered a demonstration of the system and a sample of the kind of analysis made possible thereby.

4.1 Analysis Approach: Case Studies

This analysis takes the form of three case studies packet types of particular interest – TCP/22 in Section 4.2, TCP/3389 in Section 4.3 and UDP/1434 in Section 4.4. This approach has been chosen because of the prevalence of virus and scanning activity on these ports, which are all used for services that, if compromised, give attackers access to full control of the machine and sensitive data.

A notable omission is that of TCP/445, the port targeted by the Conficker worm (Aben, 2009). It was decided not to analyse traffic aimed at this port, despite its position as the most prevalent packet type in the 196 datasets, because of the wealth of existing work around this topic, including Aben (2009); Irwin (2011, 2012, 2013b).

The three packet types studied all have destination ports within the top ten most common destination ports for each darknet in their respective protocols, as shown in Tables 4.2 and 4.3. In addition, both TCP/22 and TCP/3389 appear in the overall top ten destination ports shown in Table 4.1, which also features TCP/1433, a related service to UDP/1433.
### Table 4.1: Top 10 destination ports for each dataset, UDP and TCP traffic combined (ports which a receive a majority of UDP traffic shaded)

<table>
<thead>
<tr>
<th>Rank</th>
<th>146</th>
<th>155</th>
<th>196-A</th>
<th>196-B</th>
<th>196-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22</td>
<td>22</td>
<td>445</td>
<td>445</td>
<td>445</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>53</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>53</td>
<td>3389</td>
<td>5060</td>
<td>5060</td>
<td>5060</td>
</tr>
<tr>
<td>4</td>
<td>3389</td>
<td>5060</td>
<td>8080</td>
<td>53</td>
<td>8080</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>80</td>
<td>53</td>
<td>80</td>
<td>53</td>
</tr>
<tr>
<td>6</td>
<td>5060</td>
<td>23</td>
<td>80</td>
<td>3389</td>
<td>80</td>
</tr>
<tr>
<td>7</td>
<td>445</td>
<td>8060</td>
<td>3389</td>
<td>23</td>
<td>3389</td>
</tr>
<tr>
<td>8</td>
<td>8080</td>
<td>1433</td>
<td>23</td>
<td>8080</td>
<td>23</td>
</tr>
<tr>
<td>9</td>
<td>19</td>
<td>19</td>
<td>1433</td>
<td>1433</td>
<td>1433</td>
</tr>
<tr>
<td>10</td>
<td>1433</td>
<td>445</td>
<td>19</td>
<td>19</td>
<td>8899</td>
</tr>
</tbody>
</table>

### Table 4.2: Top 10 destination ports for each dataset

<table>
<thead>
<tr>
<th>Rank</th>
<th>146</th>
<th>155</th>
<th>196-A</th>
<th>196-B</th>
<th>196-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22</td>
<td>22</td>
<td>445</td>
<td>445</td>
<td>445</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>3389</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>3</td>
<td>3389</td>
<td>80</td>
<td>8080</td>
<td>80</td>
<td>8080</td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>23</td>
<td>80</td>
<td>3389</td>
<td>80</td>
</tr>
<tr>
<td>5</td>
<td>445</td>
<td>8080</td>
<td>3389</td>
<td>23</td>
<td>3389</td>
</tr>
<tr>
<td>6</td>
<td>8080</td>
<td>1433</td>
<td>23</td>
<td>8080</td>
<td>23</td>
</tr>
<tr>
<td>7</td>
<td>1433</td>
<td>445</td>
<td>1433</td>
<td>1433</td>
<td>1433</td>
</tr>
<tr>
<td>8</td>
<td>443</td>
<td>443</td>
<td>443</td>
<td>443</td>
<td>8899</td>
</tr>
<tr>
<td>9</td>
<td>49348</td>
<td>5900</td>
<td>5900</td>
<td>5900</td>
<td>443</td>
</tr>
<tr>
<td>10</td>
<td>5900</td>
<td>1234</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
</tbody>
</table>
4.2 Case Study of TCP/22

TCP/22 is ordinarily used for the Secure Shell (SSH) service, which gives users remote command-line access to a computer (Ylonen & Lonvick, 2006). This port is a popular target of scanning and forced entry attempts via brute force on the Internet. An attacker will scan hosts and attempt to brute-force a connection by trying to make an SSH connection with random usernames and passwords on every machine it can find. An example of the logs produced by failed attempts at this attack is shown in Listing 3.

A slightly more sophisticated approach to this attack is spoof the DNS record of the IP address performing the connection, presumably in the hope that the SSH protocol will be more lenient on a given legitimate domain. Real world evidence of this kind of attack is given in Listing 4.

TCP/Port 22 is the most popular destination port and protocol in the data recorded on both the 146 and 155 darknets and the second most popular destination port on the 196 darknets, behind the Conficker worm’s TCP/445.

Figure 4.1 shows the TCP/22 packet counts over the recording period for each darknet. From this a few points can be noted: the distributions of each of these graphs are very similar. If we group the sensors into two groups, one being 146 and 155 and the other being 196-A–C, as explained in Nkhumeleni (2014) and in Chapter 3, Section 3.1, the

<table>
<thead>
<tr>
<th>Rank</th>
<th>146</th>
<th>155</th>
<th>196-A</th>
<th>196-B</th>
<th>196-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>53</td>
<td>53</td>
<td>5060</td>
<td>5060</td>
<td>5060</td>
</tr>
<tr>
<td>2</td>
<td>5060</td>
<td>5060</td>
<td>53</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>4</td>
<td>3544</td>
<td>3544</td>
<td>3544</td>
<td>6503</td>
<td>3544</td>
</tr>
<tr>
<td>5</td>
<td>1434</td>
<td>1434</td>
<td>1434</td>
<td>3544</td>
<td>1434</td>
</tr>
<tr>
<td>6</td>
<td>161</td>
<td>6588</td>
<td>137</td>
<td>1434</td>
<td>6502</td>
</tr>
<tr>
<td>7</td>
<td>123</td>
<td>161</td>
<td>161</td>
<td>18991</td>
<td>137</td>
</tr>
<tr>
<td>8</td>
<td>137</td>
<td>123</td>
<td>123</td>
<td>137</td>
<td>161</td>
</tr>
<tr>
<td>9</td>
<td>1900</td>
<td>39455</td>
<td>1900</td>
<td>39455</td>
<td>33437</td>
</tr>
<tr>
<td>10</td>
<td>39455</td>
<td>19222</td>
<td>49153</td>
<td>161</td>
<td>123</td>
</tr>
</tbody>
</table>

Table 4.3: Top 10 destination ports for each dataset
Listing 3 Excerpt from the SSH authentication logs on a server on the Internet, showing a brute-force attack

Oct 28 11:00:39 localhost sshd[22739]: input_userauth_request: invalid user chris [preauth]
Oct 28 11:00:39 localhost sshd[22739]: Received disconnect from 5.255.86.28: 11: Bye Bye [preauth]
Oct 28 11:00:39 localhost sshd[22741]: Invalid user chris from 5.255.86.28
Oct 28 11:00:40 localhost sshd[22743]: Invalid user donna from 5.255.86.28
Oct 28 11:00:40 localhost sshd[22743]: input_userauth_request: invalid user donna [preauth]
Oct 28 11:00:40 localhost sshd[22743]: Received disconnect from 5.255.86.28: 11: Bye Bye [preauth]
Oct 28 11:00:41 localhost sshd[22757]: Invalid user helen from 5.255.86.28
Oct 28 11:00:41 localhost sshd[22757]: input_userauth_request: invalid user helen [preauth]
Oct 28 11:00:41 localhost sshd[22757]: Received disconnect from 5.255.86.28: 11: Bye Bye [preauth]

Listing 4 Excerpt from the SSH authentication logs on a server on the Internet, showing a DNS spoofing attack

Oct 29 01:47:36 localhost sshd[24000]: Address 195.230.113.7 maps to srv.hosting2.danex.net.pl, but this does not map back to the address – POSSIBLE BREAK-IN ATTEMPT!
Oct 29 01:47:36 localhost sshd[24000]: Received disconnect from 195.230.113.7: 11: Bye Bye [preauth]
Oct 29 01:47:37 localhost sshd[24002]: Address 195.230.113.7 maps to srv.hosting2.danex.net.pl, but this does not map back to the address – POSSIBLE BREAK-IN ATTEMPT!
Oct 29 01:47:37 localhost sshd[24002]: Received disconnect from 195.230.113.7: 11: Bye Bye [preauth]
Oct 29 01:47:39 localhost sshd[24011]: Received disconnect from 195.230.113.7: 11: Bye Bye [preauth]
Oct 29 01:47:39 localhost sshd[24013]: Address 195.230.113.7 maps to srv.hosting2.danex.net.pl, but this does not map back to the address – POSSIBLE BREAK-IN ATTEMPT!
4.2. CASE STUDY OF TCP/22

distributions within each group are more similar still, with the 196 sets experiencing a
spike of traffic in July 2013 that is not shown in the other group.

Further analysis reveals patterns that could be attributed to scanning activity. Figure
4.2a shows packets sent from a single IP address to destinations within the 146 darknet,
and Figures 4.2b–4.2g each show a portion of this traffic for six individual addresses.
These six graphs are all identical in shape and scale, and similar to the graph in Figure
4.2a, so it can be assumed that the activity portrayed is a result of number of regular
programmatic scans, potentially from different hosts but using the same spoofed source
IPs scans of TCP/22 over an IP range.

In order to further ascertain whether or not this traffic comes as a result of scanning
activities, the system’s colour spectrum script has been used on the data displayed in
Figure 4.2. The result is shown as Figure 4.3. This spectrum indicates repeated scanning
of hosts within the darknet in a largely erratic order but in ascending order of final octet
values in some areas. The addresses scanned do not have uniform gaps between their final
octet values, and within the range there exist numerous concentrated bands of very close
addresses. As it is not logical for a single host to send packets to multiple destinations in a
single scan and yet this spectrum contains repeated colours, it is likely that the source IP
address in these packets been spoofed and the scans are coming from multiple different,
likely compromised hosts.
(a) TCP/22 traffic from 22-1 to destinations within the 155 darknet

(b) TCP/22 traffic from 22-1 to a single destination (22-A) within the darknet

(c) TCP/22 traffic from 22-1 to a single destination (22-B) within the 155 darknet

(d) TCP/22 traffic from 22-1 to a single destination (22-C) within the darknet

(e) TCP/22 traffic from 22-1 to a single destination (22-D) within the 155 darknet

(f) TCP/22 traffic from 22-1 to a single destination (22-E) within the darknet

(g) TCP/22 traffic from 22-1 to a single destination (22-F) within the 155 darknet

Figure 4.2: Scanning graphs of TCP/22 traffic from a single source IP (22-1) both overall and for six individual destination addresses (22-A–F)
Figure 4.3: Colour spectrum of the earliest captured 1000 TCP/22 packets sent from source address A to addresses within the 146 darknet
4.3 Case Study of TCP/3389

TCP/3389 is the port ordinarily used by the Microsoft Windows Remote Desktop Protocol, the protocol for Remote Desktop\(^1\), an application which allows users to log into and operate remote machines using a the Microsoft Windows GUI environment, although there do also exist non-Microsoft, non-Windows, cross-compatible implementations of the program\(^2\). This is in contrast to services like SSH and Telnet (Postel & Reynolds, 1983), which just give a remote user access to a text-only command prompt connection.

Irwin (2013a) conducted a case study of traffic on this port, noting that traditionally very little of this traffic type had been recorded within IBR, but this ceased to be the case in August 2011, with the advent of the Morto worm, which brute-forces the passwords of machines over Remote Desktop in order to gain control over them. This worm is extensively documented by Bitton (2011).

As Figure 4.4 shows, activity captured on this port can be likened between darknet sensors. Darknets 146 and 196-A both show a relatively stable, low level of traffic on the port with a single large spike occurring in January 2014 – on the 5th of January in the 146 darknet and on the 18th of January 2014 in the 196-A darknet. The other darknets show the same general traffic pattern, but without the spikes.

\(^1\)http://support.microsoft.com/kb/306759

\(^2\)http://www.rdesktop.org/
The TCP/3389 traffic does not show the same scanning patterns as the TCP/22 traffic discussed in Section 4.2, with the various destination addresses receiving very few packets each from the source address. A notable exception to this pattern occurs in the 155 darknet, with traffic from a host that will be termed 3389-1. This host sent significantly more packets to each of its destinations than any other observed source. The resulting graphs are shown in Figure 4.5a and its sub-figures, which contain graphs of traffic from 3389-1 to six individual hosts within the 155 darknet, from 3389-A to 3389-F.

Figure 4.6 shows a colour spectrum of destination hosts sent packets by source address A. This spectrum shows more gradual changes over the address space with fewer large clusters of addresses than were present in the TCP/22 traffic explored in Section 4.2. As was the case with TCP/22 in Section 4.2, it is not logical for a single host to send packets to multiple destinations in a single scan and yet this spectrum contains repeated colours, so it is likely that the source IP address in these packets been spoofed and the scans are coming from multiple different, likely compromised hosts.

4.4 Case Study of UDP/1434

The TCP/1433 and UDP/1434 ports are used for communication with Microsoft SQL server. Databases are a common entry-point for attackers as they often contain sensitive data, and in addition, many SQL database implementations have features for dynamically interfacing with the operating system, which can be used by attackers to gain illegitimate access to a system. An example of this kind of feature is xp_cmdshell\(^3\), which provides Microsoft SQL Server with a way of executing system commands from within SQL statements.

Traffic on this port and protocol for all five darknets is shown in Figure 4.7. This traffic is largely homogeneous across the five darknets, excepting the spikes seen in the 146 darknet in 11 November 2013 and 26 December 2013. For comparison, the significantly more numerous traffic for the TCP alternative to this service on TCP/1433 is shown in Figure 4.8. In keeping with the general dominance of TCP over UDP within the datasets, the data captured for the TCP port is far more numerous than that captured for the UDP port.

The SQL Slammer worm, a known component of IBR that operates on this port (Wustrow et al., 2010), operates on UDP/1434. At 376 bytes, the source code for this worm is small

Figure 4.5: Scanning graphs of TCP/3389 traffic from a single source IP (3389-1) both overall and for six individual destination addresses (3389-A–F)
Figure 4.6: Colour spectrum of the earliest captured 1000 TCP/3389 packets sent from source address A to addresses within the 155 darknet
Figure 4.7: UDP/1434 for all five darknets

Figure 4.8: TCP/1433 for all five darknets
enough to fit into a single UDP packet Moore et al. (2003b). From this we can conclude
that only a single packet needs to be sent to each host in order for the worm to successfully
propagate. This behaviour is confirmed by the behaviour seen in the scanning graphs for
this port, as shown in Figure 4.9a and its sub-figures, which show traffic from a source
address termed 1434-1 to destination addresses termed 1434-A–F.

Figure 4.6 shows a colour spectrum of hosts in the darknet which received UDP/1434
packets from 1434-1. This spectrum contains smaller clusters of close together destination
IP addresses than the TCP/3389 spectrum in Section 4.3, Figure 4.6, and appears to
alternate frequently between low final IP address octets (black) and high IP address
colors (red). As was the case with TCP/22 in Section 4.2 and TCP/3389 in Section 4.3,
it is not logical for a single host to send packets to multiple destinations in a single scan
and yet this spectrum contains repeated colours, so it is likely that the source IP address
in these packets been spoofed and the scans are coming from multiple different, likely
compromised hosts.

4.5 Summary

In this chapter, case studies were made of three highly prevalent packet types within the
five network telescope data captures, these being TCP/22 in Section 4.2, TCP/3389 in
Section 4.3, and UDP/1434 in Section 4.4.

The visual similarity between graphs of overall traffic captured from a source address
and traffic captured from that source address to individual destination addresses as an
indicator of scanning activity, shown in Section 4.2 is the main discovery made in this
chapter. By applying this finding programmatically over a of packets – in other words, by
programmatically determining the visual similarity between a source IP address’s total
traffic output on a particular protocol and port and the traffic outputs for individual des-
tination addresses, this kind of SSH scanning activity can be identified in packet captures.

Another finding was the theory that because a logically carried out, deliberate scan from
a single host is not likely to scan the same hosts more than once, this traffic does not
represent such a scan, and the source addresses on the packets surveyed are very likely to
be spoofed.
(a) UDP/1434 traffic from a single source (1434-1) to destinations within the 155 darknet

(b) UDP/1434 traffic from a single source to a single destination (1434-source (1434-1) to a single destination (1434-A) within the 155 darknet

(c) UDP/1434 traffic from a single source (1434-1) to a single destination (1434-B) within the 155 darknet

(d) UDP/1434 traffic from a single source (1434-1) to a single destination (1434-C) within the 155 darknet

(e) UDP/1434 traffic from a single source (1434-1) to a single destination (1434-D) within the 155 darknet

(f) UDP/1434 traffic from a single source (1434-1) to a single destination (1434-E) within the 155 darknet

Figure 4.9: Scanning graphs of UDP/1434 traffic from a single source IP (1434-1) both overall and for six individual destination addresses (1434-A–F)
Figure 4.10: Colour spectrum of hosts within the 196-B darknet scanned by 1434-1.
Chapter 5

Conclusion

Thus far, this document has provided a general survey on existing research in the field of internet background radiation and network traffic analysis, a detailed breakdown of the system’s components and a sample analysis of results achieved with the system. In this final chapter, the document concludes by looking back at the goals stated in Chapter 1 and relating to the work done.

This conclusion is divided into four main sections. Section 5.1 contains a summary of the thesis. The research conducted is concluded in section 5.2, and finally future work in this area is discussed in section 5.3.

5.1 Research Summary

Chapter 1: following a brief passage introducing IBR, the research problem was stated in section 1.1 and the goals were described in section 1.2. The scope of the research was defined in section 1.3 and the research approach and structure of the document were outlined in section 1.5.

Chapter 2 contained the literature review, which examined previous research in the area. It detailed the discovery of IBR, the use of darknets and similar network telescopes (such as greynets) to gather data and various means of processing that data – packets, netflows and honeypots. It also looked briefly into the use of BGP tables to detect spoofed traffic within darknet data, although this was ultimately removed from the scope of the project.
5.2. RESEARCH GOALS

Chapter 3 detailed the datasets used for analysis and the tools used to create the system for performing that analysis. The sources of the datasets were given, the libraries and external tools used to create the system were enumerated. The system itself and its three main phases – data loading, data exploration and data analysis – were explained. Time and disk space metrics for the system’s operations were also provided.

Chapter 4 provides a sample analysis, demonstrating the use of the system. This involved the characterisation of some key aspects of the data through the lens of three case studies on selected significant and highly prevalent packet types. The chapter concluded by stating the value of the finding and putting the analysis conducted into a broader context.

5.2 Research Goals

The goals of this research, as given in Chapter 1, are revisited below.

To develop a proof of concept system for conducting analysis on IBR data, focusing on speed, extendability and flexibility.

The achievement of the first goal is detailed in Chapter 3. Although the system takes a longer time to load in the packet data than is ideal, once the data has been loaded and indexed the system operates sufficiently fast for in-depth and multi-faceted analysis.

More generally, this project can be considered an investigation into the viability of Elasticsearch and Kibana as tools for network traffic analysis. From the results achieved we can conclude that Elasticsearch and Kibana are well suited to this task. Elasticsearch’s focus on speed over ability to modify data is perfectly suited for packet capture analysis, as the data in individual packets does not need to be changed. Kibana’s dynamic, configurable interface makes it easy for an analyst to drill down through the data and look at different facets rapidly.

A downside of using Kibana for this analysis found was Kibana’s incompatibility with Elasticsearch’s aggregations and thus inability to take advantage of the range aggregations for the area-relevant IP address type. An avenue for potentially remedying this are discussed in Future Work (Section 5.3).

To demonstrate the use of this system on sample data sourced from Rhodes University’s five network telescopes.
5.3. FUTURE WORK

To identify trends in the data that can provide the basis for further extensions to the system and future work in the research area.

The second and third goals were achieved in the construction of Chapter 4, which functions as a demonstration of the use of the system and highlights the determining of similarity between graphs of traffic sent from the same (potentially spoofed) source address to different destination addresses as an area in which further extensions to the system can be attempted.

In summary, the work done in this thesis signifies the following:

- A demonstration of the utility of visualisation through graphs and other means as necessary for human analysis of packet data, and of the potential for this analysis to contribute towards programmatic analysis of the data.

- An investigation concluding that Elasticsearch and Kibana are useful and relevant tools for investigating and analysing large packet captures.

The system constructed is open source and makes use of open source libraries and tools. Abbreviated code listings for the scripts produced and links to the public repositories are provided in Appendix B.

5.3 Future Work

The system created in this project was a proof of concept and thus contains a relatively small number of features and would be ideal for expanding and incorporating into still larger and more robust systems. Some potential expansions to the system’s functionality are given below:

- The system’s packet indexing could be improved by the storing of other packet details such as length and time to live (TTL). Ideally all of the recorded packet header data and potentially even packet body data could be stored in the system’s Elasticsearch index.

- Another possible extension to the system could be a programmatic way of analysing the visualised graph similarity detailed in Chapter 4, especially the section on TCP/22 (Section 4.2).
5.3. FUTURE WORK

- A relatively trivial improvement on the work done in this project would be to implement a version of the script used to load the PCAP files into an Elasticsearch instance using a more efficient tool than the Python Scapy library – dpkt or pcapy may be suitable for the task, or it could be done using C code and TCPdump.

- A beta version of Kibana 4 was released towards the end of this project’s timeframe\(^1\), featuring support for Elasticsearch’s aggregation queries and more powerful graphing capabilities. Once the software becomes stable, it may be beneficial to integrate it into the system.

More generally, further work in the field of IBR packet analysis could investigate the following:

- Practical work can be done on using BGP tables to detect spoofed data within darknet captures. This subject was briefly touched on in Chapter 2, Section 2.3, but time and scope constraints prevented any related work from being incorporated into this project.

- Another potential avenue for further work is the mathematical generation of similar data to that characterised in this thesis and by the system. Using the profile and characteristics of this data (especially the IP space colour spectrums), models could potentially be developed to generate similar but not identical data that could be said to be typical of a darknet and used for further study of IBR.

- Another important area is the development of near real-time tools for darknet analysis, to aid network administrators using darknet or greynet data in preventing attacks on their non-dark networks. The work performed here can be further abstracted from and possibly automated, potentially through the use of an artificial intelligence system. Ultimately, a system that monitors a packet sensor in real-time and provides high-level, fully parsed information that is useful and relevant to network administrator can be worked towards.

In conclusion, there exists a rich field of opportunity for further study related to this work and the field it forms part of. The contribution made by this project will aid in making this further work faster, more in-depth and more accessible.

\(^1\)http://www.elasticsearch.org/blog/kibana-4-beta-1-released/
References


REFERENCES


Irwin, B. 2013a (June). A baseline study of potentially malicious activity across five network telescopes. *Pages 1–17 of: Cyber Conflict (CyCon), 2013 5th International Conference on.*


Moore, David, Paxson, Vern, Savage, Stefan, Shannon, Colleen, Staniford, Stuart, & Weaver, Nicholas. 2003b. The spread of the sapphire/slammer worm.


Appendix A

Elasticsearch Index Structure

The JSON-formatted Elasticsearch index mapping for the Elasticsearch index used in the system is given in Listing 5. "default" is a special type that assigns a mapping that will be automatically applied to any new types created in the absence of more specific mappings for these types given on creation.
Listing 5 JSON-encoded structure of the Elasticsearch "packets" index used to store the data

```json
{
    "packets": {
        "mappings": {
            "_default_": {
                "properties": {
                    "date": {
                        "type": "date",
                        "format": "yyyy-MM-ddHH:mm:ss.SSSSSS"
                    },
                    "destination_address": {
                        "type": "string",
                        "index": "not_analyzed"
                    },
                    "destination_port": {
                        "type": "integer"
                    },
                    "protocol": {
                        "type": "string"
                    },
                    "source_address": {
                        "type": "string",
                        "index": "not_analyzed"
                    },
                    "source_darknet": {
                        "type": "string",
                        "index": "not_analyzed"
                    },
                    "source_port": {
                        "type": "integer"
                    }
                }
            }
        }
    }
}
```

An alternate form is shown in Listing 6.
Listing 6 Alternate JSON-encoded structure of the Elasticsearch “packets” index used to store the data

```json
{
    "packets": {
        "mappings": {
            "_default_": {
                "properties": {
                    "date": {
                        "type": "date",
                        "format": "yyyy-MM-dd HH:mm:ss.SSSSSS"
                    },
                    "destination_address": {
                        "type": "ip"
                    },
                    "destination_port": {
                        "type": "integer"
                    },
                    "protocol": {
                        "type": "string"
                    },
                    "source_address": {
                        "type": "ip"
                    },
                    "source_darknet": {
                        "type": "string"
                    },
                    "source_port": {
                        "type": "integer"
                    }
                }
            }
        }
    }
}
```
Appendix B

System Code Listings

Below are the four core scripts that make up the system. The private methods and public method contents have been removed owing to clarity and space concerns, but the full scripts are publically available as a Git repository\(^1\). They are all runnable with Python 2.7 and the requisite libraries.

Listing 7 loadpcap.py

```python
#!/usr/bin/python

# Usage: ./to_elasticsearch.py <packet file> <number of packets> <server to stream to> <darknet name>

def create_index():
    """
    Create index for loading packet data into.
    """

def stream_to_es(packets, server):
    """
    Process pcap data and send it to Elasticsearch
    """
    packets -- scapy packets object
    server -- Elasticsearch server URL

if __name__ == '__main__':
    https://bitbucket.org/DavidYates/ibr-system
```

\(^1\)https://bitbucket.org/DavidYates/ibr-system
class ESInterface:
    
    Object for querying Elasticsearch
    
    def __init__(self, url="localhost:9200", index="packets"):
        Instantiate ESInterface

        url -- the Elasticsearch instance (defaults to localhost)
        index -- the index to work with (defaults to the packets index)

    def search(self, fil={ "match_all": {} }, size=10):
        Build and run a search for top terms based on a given filter.

        fil -- a dictionary filter of the form { "field1":"value1" ... "fieldn":"valuen" } Multiple acceptable values for a single field can be given in a list.

    def date_histogram(self, fil={ "match_all": {} }, interval="day"):
        Get a set of buckets to create a date histogram with

        fil -- a dictionary filter of the form { "field1":"value1" ... "fieldn":"valuen" } Multiple acceptable values for a single field can be given in a list.
        interval -- the bucket interval (day, week, minute, etc)

    def top_terms(self, field, fil={ "match_all": {} }, size=10):
        Find the top terms in a given field

        field -- a field in the elasticsearch index documents
        fil -- a dictionary filter of the form { "field1":"value1" ... "fieldn":"valuen" } Multiple acceptable values for a single field can be given in a list.
        size -- the number of results to return
Listing 9 graph.py

```python
#!/usr/bin/python

class Graph:
    
    ""
    Object for drawing graphs
    ""

@staticmethod
def save_graph(date_histograms, filename="graph.png", legend=[]):
    ""
    Save Matplotlib graphs of date histograms from elasticsearch
    date_histogram -- the date histogram or list of date date
    histograms (produced by an ESInterface) to graph
    legend -- list of string names for the legend (defaults to an
    empty list and no legend)
    ""

if __name__ == "__main__":
```

Listing 10 spectrum.py

```python
#!/usr/bin/python

def draw_dport_spectrum(esi, dport, protocol, darknet):
    ""
    Draw a spectrum for destination address packets received on a given
    port
    esi -- ESInterface object
    dport -- destination port to look at
    protocol -- packet type to look at (UDP or TCP)
    darknet -- darknet to look at
    ""

if __name__ == "__main__":
```
Listing 11 scanning.py

```python
#!/usr/bin/python

def draw_scanning_graphs(esi, dport, protocol, darknet):
    """
    Draw graphs to determine the likelihood of scanning activity
    esi -- an ESInterface to use
    dport -- the destination port to look at
    protocol -- packet type to look at (TCP or UDP)
    darknet -- the darknet to look at
    """

if __name__ == "__main__":
```