ANGLIA RUSKIN UNIVERSITY

FACULTY OF SCIENCE AND TECHNOLOGY

A BOTNET NEEDLE IN A VIRTUAL HAYSTACK

MARK GRAHAM

A thesis in partial fulfilment of the
requirements of Anglia Ruskin University
for the degree of Doctor of Philosophy

Submitted: June 2017

Acknowledgements

This dissertation was prepared in part fulfilment of the requirements of the degree of Doctor of Philosophy under the supervision of Adrian Winckles and Dr Erika Sanchez-Velazquez at Anglia Ruskin University.

This Ph.D. journey would not have been possible without the support that I have received from many people. In particular, I express huge gratitude to my first supervisor Adrian Winckles for his inspiration and support. Adrian has been a mentor to me for many years. A huge thank you also to my second supervisor Dr. Erika Sanchez for her encouragement and motivation. I also extend my thanks to Chris Holmes for his friendship and companionship during my years spent at Anglia Ruskin University.

I gratefully acknowledge the funding I received for my Ph.D. from Anglia Ruskin University. Sincere thanks goes to my head of department, Professor Marcian Cirstea for his guidance and advice. May I express my thanks to other members of the department, especially my fellow Ph.D. students; Mohamed Kettouch and Dr. Arooj Fatima.

This work is dedicated to Samantha who is always there to listen.

ANGLIA RUSKIN UNIVERSITY

ABSTRACT

FACULTY OF SCIENCE AND TECHNOLOGY

DOCTOR OF PHILOSOPHY

A BOTNET NEEDLE IN A VIRTUAL HAYSTACK

MARK GRAHAM

JUNE 2017

The Cloud Security Alliance's 2015 *Cloud Adoption Practices and Priorities Survey* reports that 73% of global IT professionals cite security as the top challenge holding back cloud services adoption. Malware with the capabilities to jump between the abstracted virtual infrastructures found within cloud service provider networks heightens the threat from botnet attack upon a cloud infrastructure. This research project aimed to provide a novel methodological approach for capturing communication traffic between botnets. The originality of this study comes from the application of standards-based IPFIX flow export protocol as a traffic capture mechanism.

The first contribution to knowledge is a critical investigation into how IPFIX export overcomes the limitations of traditional NetFlow-based botnet communication traffic capture in cloud provider networks. The second contribution is the BotProbe IPFIX template, comprising eleven IANA IPFIX information elements. Field occupancy count and Spearman's Rank correlation on 25 million botnet flows created an IPFIX template tailored specifically for botnet traffic capture. The third contribution is BotStack, a modular, non-intrusive IPFIX monitoring framework, created upon Xen hypervisor and virtual switched platforms, to incorporate IPFIX export into existing cloud stacks. The fourth contribution is compelling empirical evidence from weighted-factor observation across multiple network vantage points, that siting IPFIX exporters on the host hypervisor provides maximum traffic visibility.

BotProbe performs on average 26.73%±0.03% quicker than traditional NetFlow v5, with 14.06%±0.01% less storage requirements. BotProbe can be extended with additional application layer attributes, for use in less privacy sensitive environments. Both novel IPFIX templates were tested on the BotStack framework, capturing four distinct traffic profiles in the life cycle of a Zeus botnet.

The techniques developed in this research can be repurposed to create IPFIX traffic capture templates for most Cybersecurity threats, including DDoS and spam, turning behavioural-based traffic capture from a big data challenge into a manageable data solution.

Keywords: *botnet detection, cloud service provider, ipfix, netflow, traffic capture*

Contents

List of Figures

List of Tables

List of Acronyms

AV	**A**nti-**V**irus
C&C	**C**ommand and **C**ontrol
CSA	**C**loud **S**ecurity **A**lliance
CSP	**C**loud **S**ervice **P**rovider
DDoS	**D**istributed **D**enial **o**f **S**ervice
DNS	**D**omain **N**ame **S**ystem
EE	**E**nterprise **E**lement
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
IANA	**I**nternet **A**ssigned **N**umbers **A**uthority
ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol
IDS	**I**ntrusion **D**etection **S**ystem
IE	**I**nformation **E**lement
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IPFIX	**IP F**low **I**nformation **E**xport
IPS	**I**ntrusion **P**revention **S**ystem
ISP	**I**nternet **S**ervice **P**rovider
IoT	**I**nternet **o**f **T**hings
IRC	**I**nternet **R**elay **C**hat
K-S	**K**olmogorov-**S**mirnov
MITM	**M**an-**I**n-**T**he-**M**iddle
P2P	**P**eer-to-**P**eer
PCAP	**P**acket **CAP**ture
PDU	**P**rotocol **D**ata **U**nit
PSAMP	**P**acket **SAMP**ling
SCAN	**S**mart **C**ity **A**rea **N**etwork
SCTP	**S**tream **C**ontrol **T**ransmission **P**rotocol
SMTP	**S**imple **M**ail **T**ransfer **P**rotocol
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol
SSL	**S**ecure **S**ocket **L**ayer
ToS	**T**ype **o**f **S**ervice
VE	**V**irtual **E**nvironment
VM	**V**irtual **M**achine
YAF	**Y**et **A**nother **F**lowmeter

List of Publications

Some of the ideas for the text and figures in this thesis have previously appeared in the following peer reviewed publications and presentations:

Chapter 2: Technological Review

- Graham, M., Winckles, A. 2014. An Analysis of Pre-Infection Detection Techniques for Botnets and other Malware. In: *7th International Conference on Cybercrime Forensics Education and Training*. Canterbury, UK, 10-11 May 2014. CFET.
- Graham, M., 2014. Cloud-Based Detection Techniques for Botnets and Other Malware. In: *OWASP Appsec EU 2014*. Cambridge, UK, 4 July 2014. [video online] Available at: <http://www.youtube.com/watch?v=fV5kED7nryw>.

Chapter 3: Flow Export Overview

- Graham, M., Winckles, A. and Sanchez, E., 2015b. Practical Experiences of Building an IPFIX Based Open Source Botnet Detector. *The Journal on Cybercrime & Digital Investigations*. 1(1), pp.21-28.

Chapter 4: BotProbe: A Novel IPFIX Template for Botnet Traffic Capture

- Graham, M., 2017. BotProbe: Reducing Big Data Challenges in Threat Detection. In: *CyberUK 2017*. Liverpool, UK, 14 – 16 March 2017. [poster & demonstration].
- Graham, M., 2017. BotProbe: Making Network Big Data Manageable. In*: CRESTCon and IISP Congress Conference*. London, UK, 19 April 2017. [poster & demonstration].
- Graham, M., 2017. Botnet Observations: What can be done to up our game against this significant threat? In*: Institute of Information Security Professionals (IISP) East Anglia Chapter, 2017*. Ipswich, UK, 25 May 2017.

Chapter 5: BotStack: A Novel IPFIX Framework

- Graham, M., Winckles, A. and Moore, A., 2014. Botnet Detection in Virtual Environments using NetFlow. In: *7th International Conference on Cybercrime Forensics Education and Training*. Canterbury, UK, 10-11 May 2014. CFET.
- Graham, M., Winckles, A. and Sanchez, E., 2015a. Botnet Detection within Cloud Server Provider Networks using Flow Protocols. In: *IEEE 13th International Conference on Industrial Informatics*. Cambridge, UK, 22-24 July 2015. IEEE.

Copyright

I think computer viruses should count as life. Maybe it says something about human nature, that the only form of life we have created so far is purely destructive.

Stephen Hawking, 1994

1 Introduction

The European Union Agency for Network and Information Security claims that botnets are the main component in cybercrime consumerisation and the first Cybercrime as a Service to reach maturity (*ENISA, 2015*). To entirely eliminate a botnet the Command and Control (C&C) server must be taken off-line. Forensic techniques like signature-based detection are widely used in anti-malware software. This approach can disinfect a host device against a known malware strain, but does little to aid the takedown of a botnet C&C server. Internet Service Providers are able to facilitate takedown of botnet C&C servers across the Internet through analysis of Internet traffic protocols such as DNS, to sink hole malicious IP address ranges. In a networked environment, such as a Cloud Service Provider (CSP) infrastructure, such Internet protocols may not be present. Instead, analysis of network traffic can be used to identify malicious behaviour.

Academics have constructed many botnet detection algorithms based upon traffic analysis techniques. Network traffic is sampled at multiple collection points distributed across a network. Captured traffic is then cleansed and filtered before being fed into a botnet detection algorithm. Existing traffic capture methods have distinct drawbacks. Traffic capture using flow export protocols, such as NetFlow, are limited in the traffic attributes they capture, which in turn imposes limitations upon the attributes that can be used in the detection algorithms. Where flow export protocols are not able to capture a desired traffic attribute, traffic capture is supplemented with packet capture (PCAP) (*Sperotto, et al., 2010*). In high-speed data networks, the high data volumes captured by such methods turn threat intelligence into a big data challenge. NetFlow export traffic can be one of the largest heterogeneous data sources in high-speed data networks. It can grow to tens of terabytes of data per day and is expected to grow to petabytes over the years (*Santos, 2016*).

This research project conceptually developed a novel methodological approach to botnet traffic capture using IPFIX, a new generation of flow export protocols, to address a gap in knowledge around botnet detection in CSP networks. The design, construction and validation of this approach are described within this thesis.

This first chapter introduces the reader to the motivation behind botnet detection within CSPs. The boundaries of this research are defined, before outlining the overall research approach. Finally, this chapter declares the original contributions to knowledge from this research.

1.1 Motivation

History recalls countless technologies developed to steal or injure in order to obtain other people's resources, be it land, possessions or information. Today's technologically advanced civilisation is no different; stealthy malicious software (malware) is a tool to steal information or disrupt information systems. In 1948, John von Neumann predicted that computer code will one day have the ability to reproduce itself. The essay, the Theory of Self-Reproducing Automata (v*on Neumann and Burks, 1966*), is considered by many as the forerunner theory behind today's computer viruses. In 1971, Creeper became the first self-replicating program, designed as an academic experiment to infect computers connected to ARPANET, an early version of the Internet. Thirteen years later, Cohen coined the term virus to refer to a self-replicating program, hypothesising that these would become a major computer security problem. Cohen went on to suggest that since "prevention of computer viruses may be infeasible if widespread sharing [of programs] is desired" a cure "depends on the ability to detect a virus and overcome it." (*Cohen, 1984*).

A bot is a software application that is used to perform repetitive operations - such as Google's web crawler. A bot only becomes malicious when it is used as a delivery mechanism for a nefarious payload, such as malware. A botnet is a network of infected host machines under the control of a human operator known as a botmaster. What makes a botnet different to other malware is the use of C&C channels which a botmaster uses to disseminate commands to their bot armies. Chapter 2 explains how the power of botnets comes from this critical-mass of networked machines working together as a single platform from which to launch massive coordinated attacks such as distributed denial-of-service (DDoS) and click-fraud.

Anyone can pay to gain access to a cloud service. Therefore anyone with the right motivation can attack the cloud from within. As CSPs become vital building blocks in

the Internet of Things (IoT), as a convenient way to centrally store sensor or device data, the attack profile upon CSPs will change. Attacks have been witnessed upon the hypervisors that lie at the heart of CSP network infrastructures. Since 2014, attacks using botnets hosted within unsecure IoT devices have risen. With the public release of the Mirai bot source code in 2016 (*Mansfield-Devine, 2016*), copycat attacks have started to increase.

The CSP built environment poses challenges for botnet detection. Privacy expectations around tenant data restrict a CSP from providing malware detection services that require packet inspection-based forensics. Furthermore, if a CSP were to locate a detection probe within the tenant environment, it could raise concerns over whether data collected for malware detection is also being surveilled. Internet Service Providers (ISPs) use DNS record analysis to takedown botnets on the Internet. However, virtualisation methods used for tenant isolation make a CSP environment closer to a traditional LAN rather than the Internet, thereby limiting the availability of DNS for botnet detection. Additionally, high-performance remains a priority in multi-tenant cloud environments from users demanding access to real-time and interactive applications and services (*Garcia-Valls, Cucinotta and Lu, 2014*).

Behaviour-based botnet detection has focused on NetFlow v5 as the primary method of traffic capture in botnet detection. NetFlow is an old protocol and has its drawbacks. Early flow protocols, such as NetFlow, were created to capture network traffic statistics as a context for measurement, billing and network management (*Santos, 2016*). *Gates, et al.,* (2004) stated that the main weakness of NetFlow v5 for threat detection is its fixed template, which captures a rigid set of data fields that are not used in their entirety in security analysis, resulting in wasted data capture. In 2013, IP Flow Information Export (IPFIX) was created under RFC-7011 (*Internet Engineering Task Force, 2013a*) as a standards-based replacement to overcome the many shortfalls of NetFlow. Figure 1 illustrates how a gap in knowledge exists in understanding how the functionality of IPFIX can be applied to threat detection and in particular for botnet detection within CSPs.

During the course of this research, informal conversations with Amazon and Microsoft have alluded to botnet detection in privacy sensitive environments being a problem that CSPs are currently facing. Such discussions have indicated that the impact of this research is not limited to just CSPs. The contributions from this research project have impact for any data network infrastructure that has potential for botnet attack, including the IoT, Smart City Area Networks (SCANs), future home networks and corporate networks.

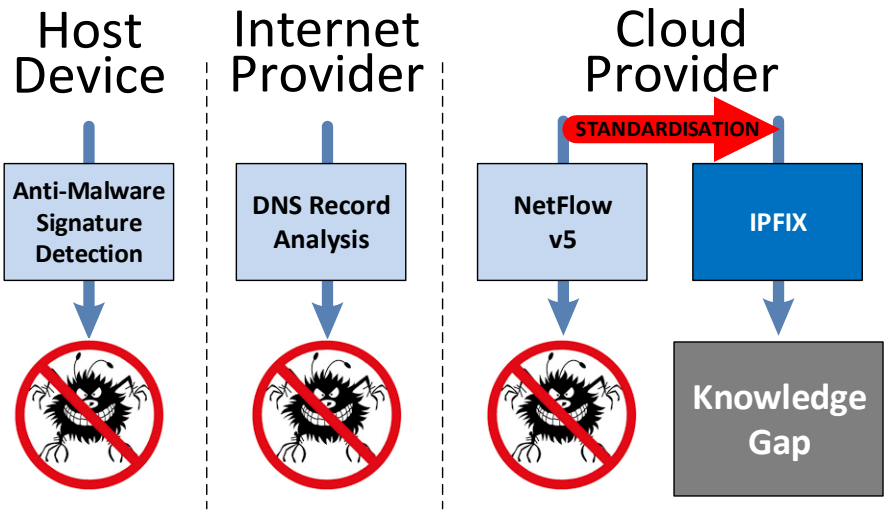


Figure 1. An illustration of the gap in knowledge in botnet detection.
Note that the gap in knowledge lies in how IPFIX can be applied to botnet detection.

1.2 Research Aim

This research has been driven by the gap in knowledge around how IPFIX can be applied to the challenge of botnet detection. The overall aim of this research is to address the problem of botnet detection in CSP multi-tenant virtualised networks through the creation of a traffic capture method utilising the IPFIX protocol. In high-speed data networks analysis of captured traffic can become a big data challenge due to traffic volumes. The proposed capture method will use features of the IPFIX export protocol to collect only network traffic pertaining to botnet communication to reduce data capture volumes, whilst retaining CSP tenant isolation and tenant privacy.

1.3 Research Hypothesis, Objectives and Boundaries

Flow export protocols, such as NetFlow, are widely utilised within CSPs for network management reporting and statistics (*Collins, 2014*). IPFIX was designed to improve the known weaknesses of the NetFlow protocol, in particular NetFlow v5 (*Trammell, et al., 2007*).

This research project was based upon the hypothesis that:

> ***"As NetFlow is already a successful candidate for behaviour-based botnet detection; the enhancements to IPFIX should offer advantages to CSPs over NetFlow v5 for capture of botnet communications."***

To achieve the aim of creating a novel botnet traffic capture method, the following objectives were addressed:

Objective #1 was a critical investigation into IPFIX as an alternative flow export protocol to NetFlow v5, for botnet communication traffic capture in cloud provider networks;

Objective #2 was the conceptual development of an innovative IPFIX template for botnet communication traffic capture in cloud provider networks;

Objective #3 was to define, design and construct an IPFIX export framework for botnet communication traffic capture in cloud provider networks;

Objective #4 was to validate the effectiveness of the novel design by demonstrating botnet communication traffic capture in a proof of concept network built upon both the template and framework.

The boundaries of this research project will be delimited to:

- Creation of a *traffic capture* method, rather than a detection algorithm;
- Evaluation of IPFIX against *NetFlow v5*, rather than NetFlow v9. In the context of this research NetFlow v9 will be considered a non-standardised forerunner to IPFIX and will be primarily ignored;
- *CSP operating environments* where infrastructure design includes tenant isolation and tenant privacy, for example Amazon EC2, Microsoft Azure and emerging IOT platforms, rather than ISP operating environments which face a different set of challenges;
- *Open source technology* where possible, to facilitate future code modification, rather than using closed source technology.

1.4 Research Methodology

All research is based on underlying philosophical assumptions about what constitutes valid research and which research methods are appropriate for the development of knowledge in a particular field. The following section outlines the assumptions and design strategies underpinning this study.

1.4.1 Justification of Research Design

An investigatory strategy was adopted to approach this research study, in order to utilise observational analysis through quantitative methods. The experimental methodology was designed in accordance with scientific method. A hypothesis was proposed in order to allow deductive reasoning to prove this hypothesis. Appropriate research objectives, see above, were devised to link evidence collection to the theory behind the hypothesis. These research objectives were defined in order to gather primary evidence from literature, as well as to obtain empirical evidence through experimentation, in order to justify the design elements for a proof of concept botnet communication capture prototype. The prototype was constructed from these design elements, and tested to compare performance against similar state of the art studies undertaken by other authors. The analysis of empirical data from testing allowed the hypothesis to be confirmed. A quantitative research method was adopted throughout this study to permit collection of numerical data for interpretation through statistical analysis. A quantitative approach allows relationships between test variables to be expressed as relative frequencies and correlation, in order to prove the theory.

1.4.2 Rationale for the Research Approach

Extensive evidence can be seen throughout global media of the extent of the social and economic threat from malicious botnets. The literature review provided a considerable body of evidence to suggest that cloud environments and virtualised infrastructures are at a risk from vulnerabilities in the underlying technologies that form these environments. In particular, vulnerabilities in hypervisors (see Chapter 2) make cloud environments an ideal tool for the rapid creation and deployment of infected virtual machines, with cloud management software being incapable of detecting such malicious behaviour. In a cloud environment all network traffic must pass through virtual switches contained within the cloud infrastructure. Therefore it is logical to monitor such devices for malicious behaviour. *Collins, (2014*) explained

how nearly all network devices used in the creation of cloud environments support flow technologies, such as NetFlow, which are used for network management and collection of network performance statistics. Literature showed that many existing botnet detection algorithms reply on packet capture or NetFlow for the collection of data to input into their detection algorithms. Research Objective #1, was prompted by studies from *Trammell, et al., (2007*) who claim that IPFIX was designed to address the known weaknesses of the NetFlow protocol.

From the outset, the purpose of this study was to capture botnet communication within cloud provider environments. Review of the state of the art revealed that multiple botnet detection algorithms exist, with varying degrees of success dependent upon the traffic attributes collected from the network. Often these detection attributes are arbitrarily selected with little supporting evidence of their relevance to botnet detection. The challenge in threat detection is not just the detection engine, but also the speed of detection. Networks, and associated business assets, are most at risk in the between infection and detection, which is on average 191 days (*IBM, 2017*).

It became evident that the drawback of most existing traffic capture techniques, when applied to high throughput networks, is that they collect big data; large volumes of highly varying information that requires processing for insight and decision making. As network traffic volumes continue to increase year upon year, analyses time is lengthened simply from increasing data volumes. Therefore this research set out to create a more efficient method of traffic capture that can be applied to existing botnet detection algorithms (see Chapter 3). Research Objective #2 was built upon claims by *Trammell, et al. (2007*) that IPFIX overcomes weaknesses in NetFlow v5, and took inspiration from *Gates et al., (2004*) who stated that the fixed template nature of NetFlow v5 limits its suitability to security analysis. Empirical data evidence was collected to demonstrate that IPFIX collects smaller data volumes than NetFlow and PCAP, is faster and demonstrates no noticeable impact upon host device CPU consumption. With Research Objective #2 demonstrating advantages of IPFIX when applied to botnet traffic communication capture, Research Objective #3 was defined so as to demonstrate how IPFIX collection can be incorporated into a prototype cloud stack. The theory of IPFIX export of botnet communication traffic was proven by testing the prototype with IPFIX templates to capture a real world botnet deployed within a sandboxed test network. The research approach is summarised in Figure 2.

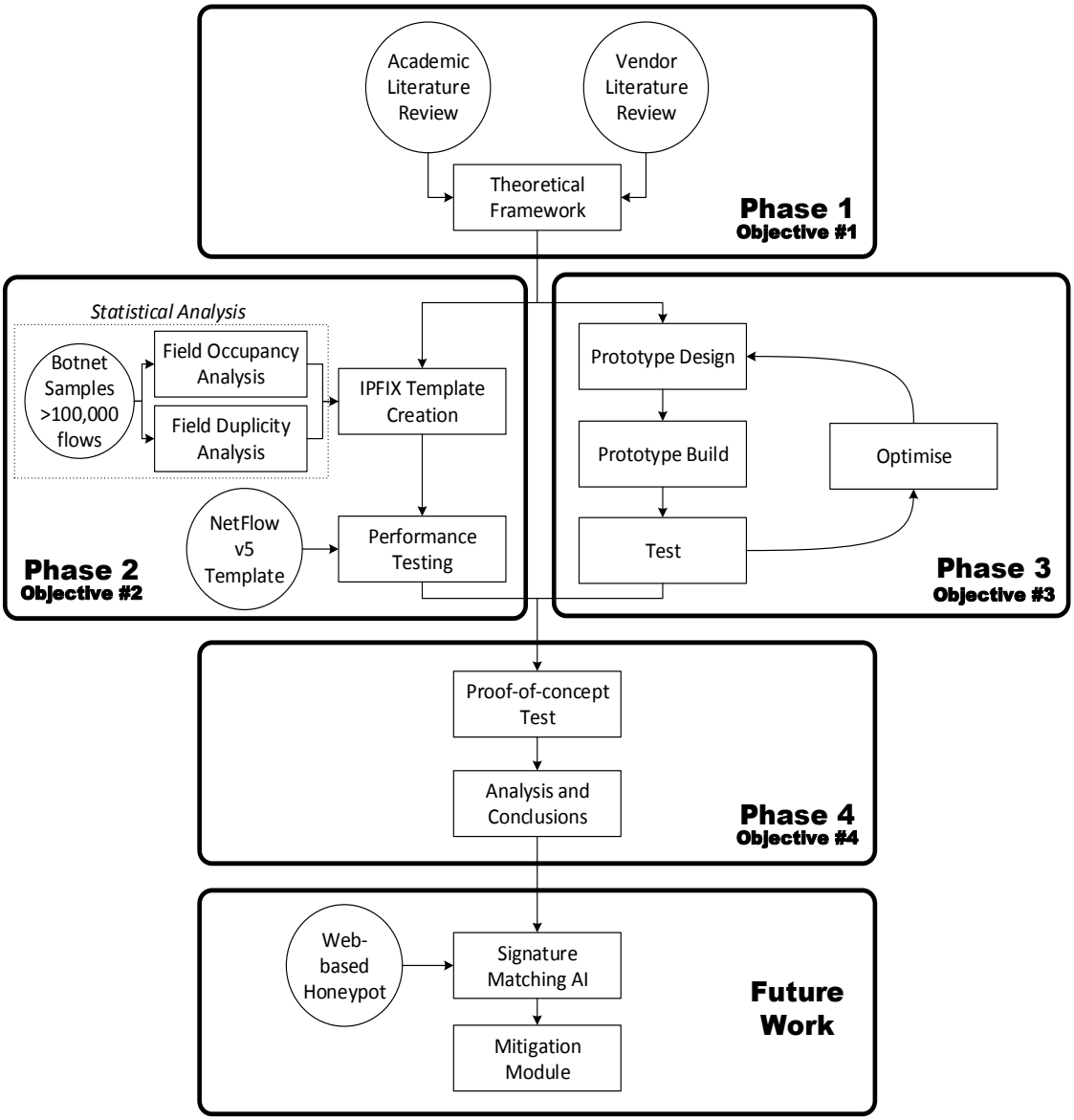


Figure 2. The phases of the research approach.

The approach to researching the problem of botnet detection in CSP networks was conducted in four phases. Phase 1 was a review of technology and state of the art in botnet detection methods. Phase 2 was the construction, justified through empirical statistical evidence, of an IPFIX template for botnet traffic capture, which was performance tested against NetFlow v5. Phase 3 was the evaluation of component elements for the construction of a modular framework architecture that allows IPFIX export to be built into cloud network stacks. Phase 4 validates the design of the template and framework in a botnet traffic capture test.

1.4.3 Experimental Design

Figure 2 shows three distinct experimental phases within the research approach. Phase 2 used statistical analysis to create the IPFIX template, and also to demonstrate advantages in performance over other capture methods. Phase 3 was the design of IPFIX export into the cloud stack. Phase 4 tested the stack and the templates against real world malware. Thus addressing Research Objectives 2, 3 and 4.

The methodological approach for each phase of experimentation is outlined within the following chapters; creation of the BotProbe templates in chapter 4.3 with performance testing in chapter 4.8, testing of the BotStack framework in chapters 5.4 and 5.6, and concept validation testing in chapter 6.3. Each methodology details the dataset origins, equipment used, methodical procedures and justification for the methods of analysis. Each chapter includes a critical discussion of the results together with the implications of how the experimental findings contribute towards the hypothesis of botnet communication traffic capture in cloud provider networks.

1.5 Ethical Considerations

All original research undertaken within Anglia Ruskin University must be performed in accordance to the university's guidance and mandatory training on ethical research. Whilst this study did not involve animal or human participation, the research does involve handling of malicious software (malware). Therefore due ethical consideration was required to protect the researcher and the reputation of Anglia Ruskin University. Throughout this study the following ethical considerations were adhered to:

- Malware was only executed in a sandboxed environment which was physical air-gap from both the university network and the Internet;
- Malware was used for testing purposes only, and not used for malicious intensions or personal gain;
- Malware was stored on appropriately labelled USB sticks, and kept in a restricted access environment;
- Malware and PCAP were only obtained through legitimate repositories;
- Where PCAP contains personal identifiable information, confidentiality was respected through the redaction of personal data.

1.6 A Summary of the Contributions to Knowledge

Figure 3 outlines how this research makes several practical as well as theoretical contributions to knowledge, indicating in which peer reviewed articles these findings have been published:

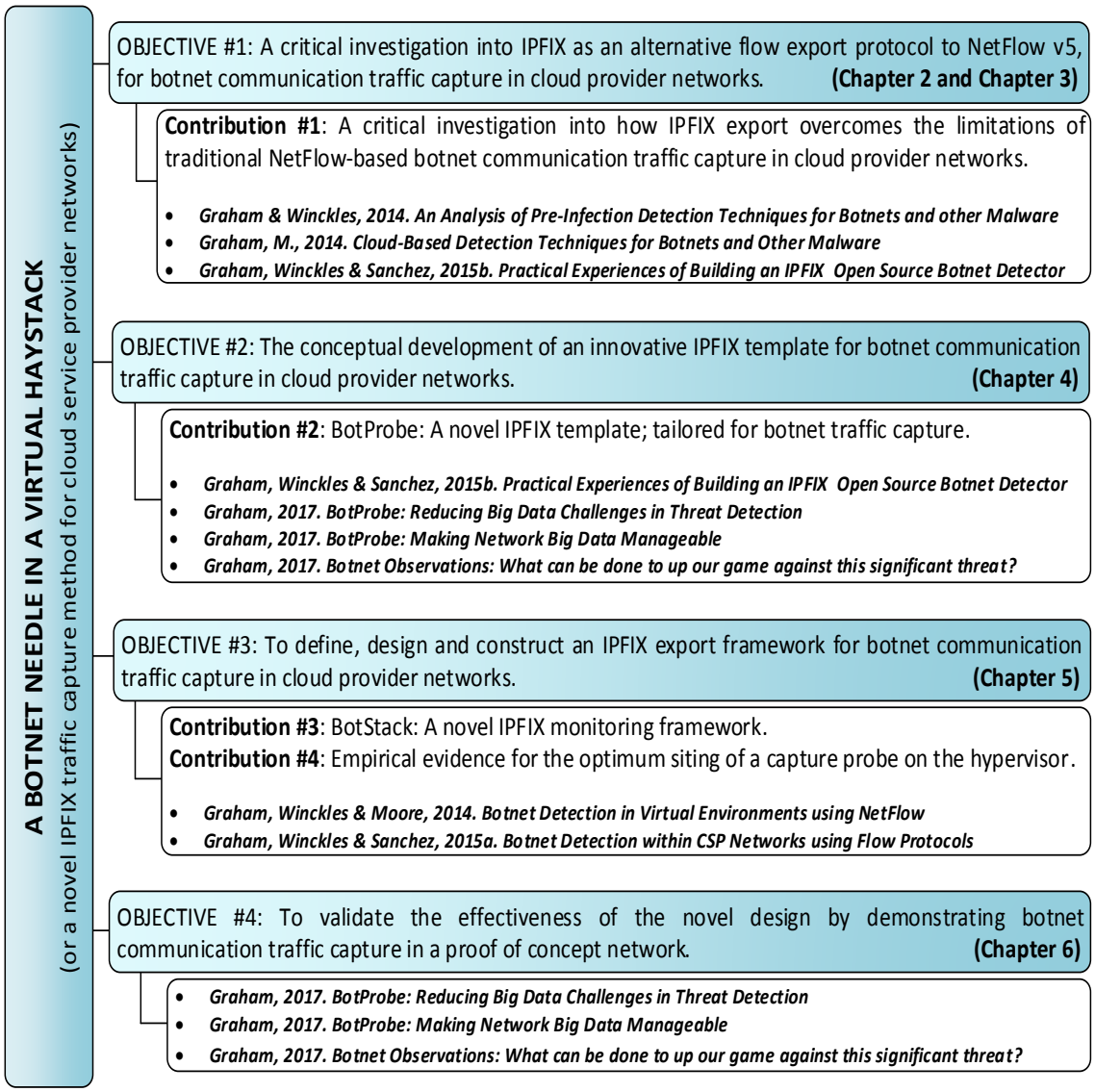


Figure 3. A diagramatical view summarising how this thesis contributes to knowledge.

1.7 Thesis Structure and Organisation

The remainder of this thesis has been structured as follows:

Chapter 2: This chapter justifies the necessity for the research within this project. Real world examples are used to describe how new families of bots are beginning to target cloud infrastructures. The chapter also reviews the legal responsibility for cloud providers towards botnet takedown.

Chapter 3: Through a thorough review of academic literature, this chapter outlines the features of IPFIX that bring advantages over NetFlow v5 to botnet traffic capture in a cloud service provider environment. The chapter reviews the state of the art in botnet detection using flow protocols, whilst explaining how IPFIX could be applied to improve these studies. NetFlow v9 is discussed, although throughout this thesis NetFlow v9 is considered to be a vendor proprietary forerunner of IPFIX and is therefore given marginal attention.

Chapter 4: Using the understanding gained from chapters 2 and 3, this chapter describes the process by which almost 25 million botnet traffic flows are analysed in the creation of the novel BotProbe IPFIX templates. Empirical data is collected and analysed for the performance impact (processing times, data volumes and CPU loadings) of the BotProbe templates against NetFlow v5.

Chapter 5: This chapter presents BotStack, a novel IPFIX framework for botnet traffic capture in cloud provider networks. Using this framework, a proof of concept network is constructed to determine the optimum siting of IPFIX sensors, as well as an understanding of how sensor clock settings impact traffic data capture.

Chapter 6: The interoperability of BotProbe with BotStack is validated by infecting a proof of concept network with the Zeus botnet. BotProbe captures botnet traffic communications across four distinct profile phases of a botnet life cycle.

Chapter 7: The final chapter summarises the research findings, drawing conclusions of the impact of this study for cloud service providers. This chapter also highlights future directions for this research.

2
Technological Review

2.1 Introduction

The aim of this research project, as described above, is to understand how the IPFIX protocol can form a capture mechanism for botnet traffic in CSP multi-tenant virtualised networks. The starting point to achieving this aim is to understand what makes a CSP infrastructure vulnerable to attack, the potential threats from malware that specifically abuses the cloud and the limitations of current detection techniques.

Many real world scenarios exist where cloud infrastructures have provided host to botnet C&C servers. The Cloud Security Alliance (CSA) recognises *Abuse and Nefarious Use of Cloud Services* as a top threat to cloud computing (*Brook, et al., 2016*). The Internet of Things (IoT) has started to become intrinsically linked to cloud hosted infrastructures, due to its ability to store data centrally. The International Data Corporation (IDC) predicts that by 2020, over 90% of all IoT data will be hosted on cloud platforms (*Turner, 2015*). Increasingly IoTs devices have been found hosting botnets; 2013 had the Darlloz botnet (*Hayashi, 2013*), 2014 had fridges sending SPAM (*Thomas, 2014*), 2015 had Lizard Stresser (*Krebs, 2015*) and 2016 had the Mirai botnet (*Mansfield-Devine, 2016*) which sustained a record-breaking 620Gbps DDoS attack, before the source code was publically released prompting copycat attacks.

The attack surface of a cloud can be extended both through infrastructure design techniques and the hardware or software upon which it is provisioned. Tenant separation is typically achieved through virtualisation of both physical (Input/Output, CPU) and networking (switching, storage) functions. Real world attacks have utilised hypervisor vulnerabilities to allow malware to jump into, across and out of virtualised environments. In a CSP this could mean an attack on another tenant, or upon a storage repository.

2.2 Botnet Fundamentals

A *bot*, or robot, is nothing more than a software application that performs repetitive operations; such as the web crawlers used by Google to index the Internet. A bot only becomes malicious when it is used as a delivery mechanism to transport nefarious payloads to multiple devices. Compromised devices work together to form a network of infected machines, or botnet. Under the remote control of a human botmaster operator, these machines lie dormant until they reach a critical mass of infected devices and the botmaster initiates an attack. Typically, an attack takes the shape of a mass attack of many hundreds of thousands of bots upon a single victim. The contender for the first botnet lies with either the Sub7 trojan, or the Pretty Park worm. Both appeared at about the same time in 1999 and both introduced the concept of connecting to victim machines via an IRC communications channel which was used to deliver malicious commands. Since then, botnets have evolved in threat, stealth and danger to become one of the most significant cybersecurity threats faced by organisations and individuals today.

Without doubt, the power of a botnet comes from this "mass" of networked machines working together as a single platform. A larger botnet is both more effective at achieving its objective, as well as harder to takedown. However, more than a few hundred thousand bots and the botnet becomes easier to detect; fewer than a hundred thousand bots and the attack becomes less effective. It is estimated that at any one time, globally as many as 2.5 million devices are connected to over 5000 C&C servers (*Trend Micro, 2016*). With botnets-for-hire services starting from $20 an hour for a DDoS attack, such services are a tool for non-specialised individual to perform powerful attacks (*ENISA, 2015*).

The ability to perform a mass attack upon a single victim, using globally distributed bots makes botnets suited for specific types of attack:

DDoS In 2015, it was estimated that malicious bots generate 29% of all Internet traffic (*Imperva, 2015*). The first half of 2016 saw more 100Gbps+ attacks than in all of 2015 (*Arbor Networks Inc., 2016*), including the first 600Gbps+ DDoS attack (*Kaspersky Lab, 2016*). 93% of DDoS attacks in Q1 2016 were from DDoS as a Service (*Incapsula, 2016*). In 2016, the Lizard Stresser botnet used IoT Webcams to launch a 400Gbps DDoS attack (*Krebs, 2015*). In 2016, a CCTV botnet was detected generating over 50,000 HTTP requests per second during a targeted DDoS attack (*Cid, 2016*).

Identity Theft Botnets targeted more than 1,500 financial institutions across more than 100 countries in 2015 (*Dell SecureWorks CTU Threat Intelligence, 2016*). During two months in 2014, a single botnet mined $600,000 in crypto coins (*Dell SecureWorks CTU Research Team, 2014*). Over 400m identities were stolen globally in 2015 from bots such as Dridex, Simba and Ramnit (*Symantec, 2016a; Verizon, 2016*).

Click Fraud It has been estimated that botnets could have cost the global advertising industry in excess of $7.2 billion in 2016 (*Association of National Advertisers and White Ops, 2016)*.

2.3 A Review of the Limitations of Botnet Detection Techniques

Amazon's AWS Security Best Practices (*Todorov and Ozkan, 2013*) recommends that tenant protection from malware is achieved through anti-virus (AV) software, anti-spam (AS) software and host-based Intrusion Detection System (IDS) software. Amazon's recommendations to protect against botnets include device patching, only using trusted software and applying the principle of least privileges. The CSA (*Cloud Security Alliance, 2011*) recommends securing the network and preventing tenant data leakage through a combination of VLANs (Virtual LANs), IDS, IPS (Intrusion Prevention System) and Firewalls. *Modi, et al.,* (2013) agree, advising on deploying IDS and/or IPS for anomaly detection within cloud architectures.

This advice has several weaknesses. Primarily, the advice puts the responsibility upon tenants to provide self-protection. Such an approach may protect individual tenants, but does not prevent the nefarious use of cloud services. A CSP hosting protection services within the tenant environment has two considerations; the privacy implications of such systems as they are based upon packet inspection techniques, and that a malicious user can disable these protection services. As the size of IoT networks increases, effective endpoint security becomes more difficult. Furthermore, AV, IDS and IPS are built upon signature-based detection engines. Signature-based detection has three drawbacks (*Graham and Winckles, 2014; Graham, 2014*): (1) an inability to cope with malware polymorphism and metamorphism, where any change to a virus binary requires a new signature definition; (2) a lack of zero day protection until a signature is created and deployed; and (3) it is a post-infection technique taking action only after malware has entered a system. Heuristic detection may overcome some of these drawbacks, but such techniques can be slow and are still subject to the same binary obfuscation techniques that malware authors deploy against signature-based

detection. Signature-based techniques do have established roles in the real-time protection of individual devices against malware. However, in a botnet of 100,000 devices, inoculating one device has little impact on the overall botnet. The takedown of a botnet requires locating and eradicating the C&C server, which is something that signature-based AV, IDS and IPS are not capable of doing.

Internet Service Providers can successfully takedown botnets by blackholing appropriate IP address ranges using DNS record analysis. In practice, DNS takedown is a slow process and can take months to successfully trace a C&C server. DNS blacklisting is susceptible to evasion techniques such as IP fluxing and domain fluxing, such as those used by the Conficker and Torpig botnets (*Graham and Winckles, 2014*). A takedown strategy is often reliant upon accurate estimations of the botnet size. DNS can be unreliable when used in botnet size estimation (*Rajab, et al., 2007*). Blackholing techniques rely upon DNS records, which are intrinsically linked to the Internet. Within a networked environment such as a CSP infrastructure, DNS is not required to obtain device addressing information, so DNS may not be present in a LAN. Thereby defeating DNS record analysis as a protection mechanism in such environments.

Every botnet uses a communications channel which enables communication between each bot and their C&C server(s). The necessity for bots to communicate with their peers throughout their life is fundamental to behaviour-based detection (*Gu, et al., 2007*). First generation botnets used a traditional client-server topology. This communication model is straightforward to set up and maintain, but fewer C&C servers make the botnet more liable to takedown. Some first generation botnets increased resilience by using primary/standby servers, but these more rigorous communication models required complex code. To increase resilience to takedown, botnets migrated to a decentralised peer-to-peer (P2P) topology where each bot node can act as a client or a server. P2P botnets display similar behaviour to benign P2P software making them more difficult to detect (*Yen and Reiter, 2010*). Out of the box P2P protocols maintain internal models of neighbouring nodes, allowing researchers to infiltrate a P2P botnet by becoming part of the botnet to obtain neighbour node IP addresses. Phatbot attempted to overcome this by using WASTE, an anonymous P2P protocol. Although WASTE does not scale for large networks (*Wang, Sparks and Zou, 2010*). The meshed ad-hoc nature of P2P networks, together with a lack of message delivery guarantee, can mean P2P botnets suffer from higher command propagation time across the network which impacts bot synchronisation (*Zhao, et al., 2013*). C&C and P2P models are both still used in botnet topology. The communication channel

protocol can also be used as an attribute in behaviour-based detection. Early bots utilised IRC. Whilst straight forward to code, IRC is easily mitigated by blocking IRC ports. A common method for achieving stealth is to employ HTTP as the communication protocol. By masquerading as legitimate HTTP or encrypted HTTPS traffic, bots cans bypass port-filtering firewalls and IDS detection (*Zeidanloo and Manaf, 2009*). IRC and HTTP are both still used as botnet communication channels.

2.4 A Review of Botnet Mitigation Legislation and Responsibility

In 2010, the OECD (Organisation for Economic Cooperation and Development) released research suggesting that global ISPs should take steps to mitigate the threat from botnet SPAM (*van Eeten, et al., 2010*). The report achieved little other than to raise an argument around where responsibilities lie for tackling botnets. In 2013, the US Federal Communications Commission (FCC) produced a voluntary code of conduct for ISPs (*Communications Security, Reliability and Interoperability Council, 2013*). The US-centric report suggested that ISPs collaborate on botnet detection and eradication. The report highlighted two hindrances preventing collaboration. First, the cost of implementing technical solutions; and second, that global laws and policy discourage global collaboration. Since its release, several major US telecommunications providers have adopted the code, including AT&T, Comcast, Sprint and Verizon. Similar voluntary codes have followed in other countries. These types of codes of conduct will remain voluntary until international law can resolve collaboration issues. Demarcation points in responsibilities are also open to interpretation. *Fryer, Stalla-Bourdillon and Chown* (2015) argue that the third party companies which host website platforms should also take some responsibility toward botnet mitigation.

In 2014, the CSA launched an anti-bot working group to coordinate research into botnet prevention within CSPs, however by 2017 this working group appeared to have become inactive having released little information. In 2015, a report from the US Federal Trade Commission (FTC) highlighted that IoT devices must implement security by design, rather than treat security as an afterthought (*Federal Trade Commission, 2015*). ISO/IEC27018:2014 is an attempt, through non-legal regulation, to address the lack of cloud computing legislation. For European CSPs, this standard links to the Article 17 of the European Commission Data Retention Directive (*EC Data Retention Directive 2006/24/EC, 2006)* which stipulates that Data Controllers within the European Economic Area who are responsible for the processing of data,

must take appropriate measures to protect personal data against accidental destruction or loss, alteration or unauthorised disclosure. This international standard covers a wide variety of subjects, but concerns itself more with protection of personally identifiable information. Ultimately, the requirements of ISO27018 are not a replacement for national or international law, nor does it specifically address the risk from botnets. By the completion of this thesis in 2017, there are no legal requirements for CSPs or ISPs to actively mitigate against botnets.

2.5 A Review of the Cloud as an Attack Platform

2.5.1 ATTACKS FROM THE CLOUD

The CSA identifies *Abuse and Nefarious Use of Cloud Services* as a top threat to cloud computing (*Brook, et al., 2016*). The report cites threats that are directly related to botnets; including DDoS attacks, email SPAM and phishing, mining for digital currency, large-scale click fraud and malicious content hosting (such as a C&C server). The report recommends that CSP customers should be permitted to monitor the health of their own cloud workspace. Whilst a dashboard-style overview of bandwidth utilisation may give clues to malicious activity, it is no more than an indicator of possible threat.

One driver in the move away from maliciously infecting victim devices, towards using service providers to host malicious VMs, is the technological advantages gained from virtualised platforms. These include performance, scalability, ease of management, lower risk of detection and stability of cloud services (*Level 3, 2015*). Another driver is cost. *Bryan and Anderson* (2010) rented 10 virtual servers on Amazon's EC2 platform at a total cost of $6 to demonstrate a cloud hosted DDoS attack. *Roth* (2011) demonstrated the power of distributed computing by renting servers on Amazon's EC2 platform at a cost of $2.10 per hr per instance. Eight instances were clustered, each trying 50,000 passwords combinations per second, taking 2 minutes to perform a 39 million word dictionary attack to brute force a victim password. *Ragan and Salazar* (2014) demonstrated how an automated service for registering free-trial CSP instances could create a botnet to mine bitcoins.

Cybercriminals are businessmen, recognising the same benefits from cloud-based services as legitimate businesses. From the point of view of a botmaster, access to considerable distributed processing power at almost negligible prices provides two opportunities. Firstly, the cloud removes the necessity and complexity of infecting a victim's device. Instead the botmaster can use a cloud provider to create a VM

instance, remove any AV or IDS, infect this VM with their own bot and clone this VM multiple times to rapidly create a sizeable botnet at little cost. Secondly, the cloud can play host to the C&C servers. If an ISP blacklists the C&C server IP addresses, the distributed architectural nature of the cloud allows for rapid re-deployment of the C&C server in a totally new geographic IP subnet.

Either of these reasons makes the cloud an effective platform from which to launch botnet attacks. One of the earliest examples was in 2009, when the Amazon's EC2 platform was used to host a Zeus C&C server as a backend alternative in case the original domain was lost (*Amazon, 2009*). Again in 2009, Amazon's EC2 platform was used to launch a massive-scale DDoS attack on Bitbucket which took Amazon 16 hours to block (*Metz, 2009*). By 2014, Amazon's AWS platform was anticipated to be hosting 41% of the world's malware (*Heimer, 2014*). In 2014, Dropbox was used to deliver the Upatre SPAM bot (*Trend Micro, 2014*). Again in 2014, Dropbox was used as a C&C server for the PlugX RAT (*Pauli, 2014*). In June 2015 a botnet that was infecting Skype sessions with adware was traced back to Amazon's AWS platform (*Osbourne, 2015*).

2.5.2 IoT ATTACKS

Public infrastructure and utilities continue to remain a high profile target. In 2010, Stuxnet was the first known worm to target the industrial SCADA (Supervisory Control and Data Acquisition) control systems used in heavy industry, by attacking Iranian nuclear centrifuges (*Kushner, 2013*). Stuxnet morphed into the industrial worms Duqu in 2011 and Flame in 2012. At around the same time, poor device security facilitated a shift from attacks on industry into attacks on the IoT. In 2012, a hacker attempted a census on the entire Internet IPv4 addressing space. To perform this the Carna botnet was created from 420,000 embedded devices that used default passwords (*Botnet, 2012*). In 2013, Linux.Darlloz attacked routers, cameras and set-top boxes that used default usernames and passwords, to create a botnet to mine crypto coins (*Hayashi, 2013*). In the same year, a botnet was supposedly created using smart fridges to perform SPAM attacks, but this has since been refuted (*Thomas, 2014*). In 2015, Lizard Stresser botnet was found performing 400Gpbs DDoS attacks from home routers with default passwords (*Krebs, 2015*). In 2016, botnets took advantage of cloud based Linux IoT devices by exploiting volatile memory vulnerabilities in ELF (Executable and Linkable Format). The first was

Linux.Routrem (*Symantec, 2016b*) also known as Remaiten. This was followed by Linux.Laubot (*Symantec, 2016c*) which specifically targeted ARM architectures.

However, both of these botnets were eclipsed by the Mirai botnet which made global news headlines after sustaining a record-breaking 620Gbps DDoS attack (*Mansfield-Devine, 2016*). Following this attack the Mirai botnet source code was publically released. The end of 2016 and early 2017 saw several copycat attacks using modified Mirai code. Whilst the IoT continues to be built on cheap devices with no security or unchanged default passwords, the IoT will remain a source from which to host attacks.

2.6 A Review of the Cloud as an Attack Surface

2.6.1 ATTACKS UPON THE CLOUD

Whilst the cloud provides an effective platform from which to host botnets, the cloud infrastructure itself is not immune to attack. Since cloud computing is no more than a combination of existing computing techniques such as virtualisation, grid computing and service-oriented computing, security issues in the cloud differ little from traditional IT solutions (*Ouedraogo, et al., 2015*). What makes the cloud different to traditional computing is (1) co-residency means that tenants often share the same network infrastructure and storage with other tenants, competitors, or even hackers; and (2) in a post Snowden era, tenants are more astute in critically evaluating a supplier's capacity for surveillance. Privacy expectations limit a CSP's ability to provide tenants with the more traditional protection solutions that are associated with packet inspection techniques, such as signature-based detection.

Alternatively, leaving malware protection to the responsibility of the tenant may broaden the cloud attack surface. Deployment of insufficient or outdated anti-virus software may permit malware capable of compromising a tenant's VM. If malware can escape a VM onto the underlying CSP network infrastructure, it has the potential to attack the CSP's storage repositories or neighbouring tenants. Unsecure cloud management interfaces are another source for malware to enter the cloud infrastructure. *Somorovsky, et al.*, (2011) exploited Amazon EC2 sessions through XSS (Cross Site Scripting) in the EC2 control software interface.

2.6.2 THE HYPERVISOR EXPOSED

A hypervisor allows emulated virtual devices to share the physical host's resources. CSPs use hypervisors to allow multiple VMs to each act as a stand-alone, tightly isolated container, thereby providing tenant isolation from other tenants. Effectively, a hypervisor is software which acts as a gatekeeper between the privileged kernel domain and the unprivileged guest domain. Exploiting vulnerabilities in this software could allow unauthorised entry or exit from this isolated sandboxed environment. *Govindavajhala and Appel* (2003) demonstrated this by using a lightbulb to raise the temperature on DRAM and SRAM chips to force memory errors. These errors allowed them to exploit both Sun and IBM VMs. Chapter 2.7 describes real world hypervisor exploits.

A hypervisor can have multiple pointers to the same memory location to allow the management of numerous concurrent virtual machines. *Rutkowsa* (2004) exploited this through four lines of code known as the Red Pill. She was able to determine whether the host was physical or virtual by analysing memory registers in the Store Interrupt Descriptor Table, which in Intel processors can be accessed by non-privileged users.

Thirteen years later, at the time this thesis was written, malware still exploits red pill type vulnerabilities to determine if it is being executed within a virtual or physical machine. With the knowledge that researchers typically study malware in a sandboxed virtual environment, malware exists that will modify its behaviour, or refuse to execute, in virtual machines to prevent researchers from understanding its behaviour. Botnets with the ability to do this include Stormbot, Agobot and Phatbot. The Dyre malware determined if it was running in a virtual environment by analysing the number of processors the environment used, under the assumption that virtual environments are usually configured with only one processor and one core to save resources (*Raff, 2015*). Rather than terminate when running on a virtual machine, malware such as Cloudburst actively seeks out virtual environments in order to exploit vulnerabilities that allow it to escape out of the VM onto the underlying infrastructure.

The CSA attributes seven of its top twelve threats to the cloud to hypervisor vulnerabilities (*Brook, et al., 2016*). As hypervisors become more feature rich they start to contain more bugs in the code. Extra security features in hypervisors can make the hypervisor more visible to attack and can often hamper bot detection (*Vaquero, Rodero-Merine and Morán, 2011*) Current practices of replicating functionality at many levels, including security features at the hypervisor and guest OS levels, create

inertia in software management which often impairs performance of the systems (*Garcia-Valls, Cucinotta and Lu, 2014*).

Fully secure isolation is still an area of study. *Rushby* (1989) proposed *true isolation*, where each VM has its own resources instead of sharing resources. Few products support this, as the x86 chip set does not have a "no sharing" concept, so not all instructions in the chip can be virtualised. The Chinese Wall concept suggests that isolation can be achieved through hosting parties that have conflict of interest on separated infrastructure. This usually means additional infrastructure which increases cost whilst reducing device utilisation. *Sailer, et al.*, (2005) proposed an isolation method on the Xen hypervisor using Access Control Modules, which may reduce the opportunity for inter-VM attacks on the same host machine, but it does not eliminate vulnerabilities in hypervisor code. VM Introspection (VMI) is a technique for the transparent real-time inspection of the operation of a VM, by directly reading the volatile memory of the running VM from the hypervisor's privileged domain. Typically, VMI research is applied to malware, but *Memarian, Conti and Leppänen* (2015) detected botnets in infected cloud VMs by searching for hidden processes and DLLs. From a CSP's point of view, VMI can be performed without the knowledge of the guest OS, thereby providing a possibility of detecting malware without a probe in the tenant environment. However, this makes it an invasive technique which may breach tenant privacy expectations. Hypervisors remain a rich source of vulnerabilities that will continue to be exploited.

2.7 A Model for Attacking a Cloud Infrastructure

Hypervisor attacks have been witnessed in the wild. When considered individually, these attacks have successfully exploit different vulnerabilities in order to jump between the physical and virtual network layers.

When considered as a combined attack, it could be possible for malware to enter the cloud infrastructure and propagate across the network. In a CSP environment, malware could then attack the CSP infrastructure, in particular storage, or attack tenants in other logical partitions. In an IoT environment, malware could compromise multiple devices or sensors. Figure 4 shows a model for malware to exploit hypervisor weaknesses using real world attacks.

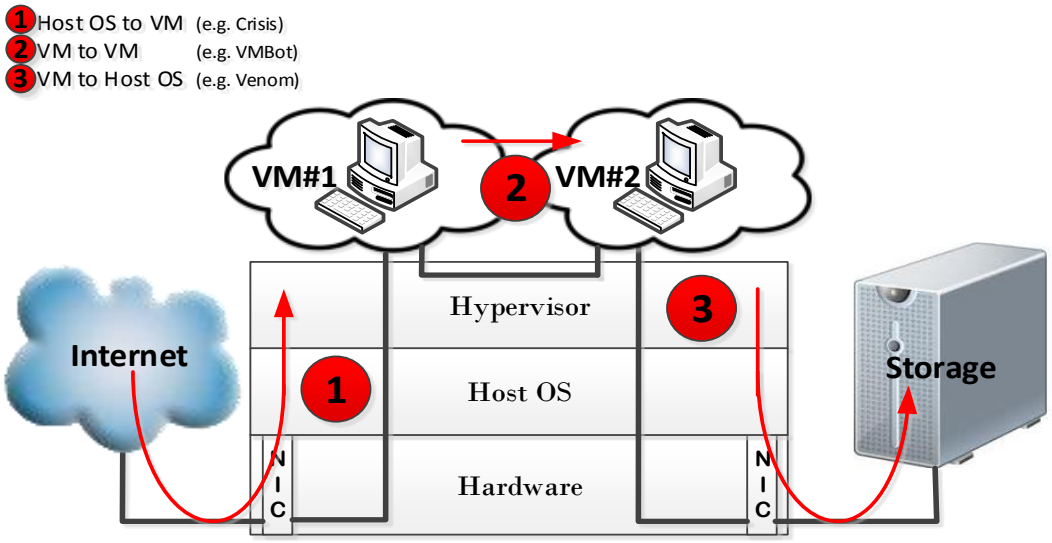


Figure 4. A three stage model for hypervisor exploitation.

The principal attack vectors, as indicated by the numbers in red circles in Figure 4, are as follows:

(1) Host Escape (Host → VM)

A hypervisor's management software could theoretically allow a cloud administrator with access to a privileged domain, access to the unprivileged guest domains. This would provide direct access to the contents of a tenant's VM memory at runtime. A CSP would mitigate this risk by applying the principle of least privileges, thereby restricting access privileges so that no single person could accumulate all the necessary privileges to do this. In August 2012, the Crisis malware was the first to perform host escape jumping from the hardware to the guest VM (*Katsuki, 2012*).

(2) VM Hopping (VM → VM)

Malware can possess the functionality to recognise when it is running in a virtual environment rather than on a physical machine. *Wang and Lee*, (2006) used memory leakage to set up covert communication channels between VMs to facilitate a cached side-channel attack on a neighbouring VM's cryptographic library. *Ristenpart, et al.*, (2009) exploited VM placement in Amazon's EC2 platform to perform side-channel attacks upon neighbouring VMs. *Vaquero, Rodero-Merine and Morán* (2011) provide a detailed study on side-channel exploitation, categorising them into three specific attack vectors: storage channel, timing channel and side-channel. *Irazoqui, et al*, (2014) captured AES keys from neighbouring VMs on Amazon's EC2 and Google

Cloud using cache leakage attacks. *Inci, et al*, (2015) repeated Irazoqui's work to capture RSA keys from neighbouring VMs on Amazon's EC2.

(3) VM Escape (VM → Host)

A larger attack surface opens up when malware can jump from the virtual environment onto the host device, and then onto the physical network. Before 2009, it was theorised that VM escape attacks could be perfromed through system directory traversal. In 2009, Cloudburst exploited corrupted memory locations in VMware shared folders to tunnel through to the underlying OS (*Kortchinsky, 2009*). In 2012, vulnerability *CVE-2012-0217* allowed an attacker to escape from a VM onto the device kernel whilst achieving escalated privileges. In 2014, Google's App Engine was exploited to allow code on the underlying OS to be executed from a Java VM (*Constantin, 2014*) and firmware bootscript vulnerabilities were found to permit guest escalation to the host (*Wojtczuk and Kallenberg, 2014*). In 2015, the Venom vulnerability (*CVE-2015-2456*) was discovered in the QEMU floppy disk controller allowing VM escape on products that had spawned from QEMU; namely KVM, Xen and VirtualBox.

QEMU vulnerabilities are the root cause of many VM escape techniques. A Xen VM guest with access to the PCNET controller can use a buffer overflow to execute packets on the host (*CVE-2015-3209*). A Xen guest with access to an emulated CDROM device has access to the QEMU process (*CVE-2015-5154*). A guest may be able to read host-level data residing in the QEMU process (*CVE-2015-5165*). Issues in unplugging an emulated block device allowed guest access to the host by unplugging the device again (*CVE-2015-5166*). A vulnerability in VMware printer virtualisation, which is installed by default, allows a guest OS to access and print documents on the Host OS. Even without VMware tools installed, the guest can still talk to the host over COM1 (*Kortchinsky, 2015*). 2015 ended in a low for QEMU with the following vulnerabilities that allowed VM escapes: *CVE-2015-5307* which also allowed the guest to DoS the host, *CVE-2015-6654*, *CVE-2015-7835*, *CVE-2015-8104* and *CVE-2015-8615*. In July 2016, a bug in Xen hypervisor (*CVE-2016-6258*), dubbed the bunker buster, allows a malicious admin within a para-virtualised VM guest to access the host via fast-paths in the page table and obtain root access. More detailed descriptions of all the CVEs mentioned above can be found at https://cve.mitre.org.

2.8 Summary

This chapter presented evidence on the reality of botnet threats upon CSPs. Fundamental exploits exist in hypervisors at the heart of CSPs making them vulnerable to attack from malware such as Crisis and Venom. Should these malwares enter a CSP environment, for example via a client's virtual machine, these exploits present other tenants and the cloud infrastructure itself as potential attack surfaces. Multi-tenancy, achieved through hypervisor technology, introduces a further challenge in creating a detector in the optimal location of a minimal number of probes to achieve maximum network visibility. This is considered in greater detail in chapter 5.

The ability to provision centralised storage for multiple distributed devices position CSPs as a vital building block for the IoT. However, there remains no reliable method for botnet detection within CSP infrastructures. AV is not the solution for botnet detection as it is incapable of tracing the C&C servers and requires the device to be compromised before detection can take place. Whereas botnets should be detected before an attack commences. Furthermore, CSPs are under no legislation to tackle the botnet threat. This differs somewhat with ISPs, who have voluntary legislation to tackle botnets, and may go towards explaining why ISPs have various solutions for botnet detection. ISP solutions generally rely upon DNS record analysis, which cannot be re-engineered for CSPs as DNS is an Internet protocol and not a local area network protocol.

This chapter begins to address research objective #1 through understanding the threats to CSPs from botnets and limitations of current malware detection in such an environment. Flow protocols are widely utilised by CSPs to capture network traffic management and reporting statistics (*Steinberger, et al., 2013).* The next chapter further addresses this research objective in considering the application of flow export within a botnet detector; focusing on the advantages of standards-based IPFIX over earlier proprietary NetFlow protocols.

3 An Overview of Flow Export

3.1 Introduction

Chapter 2 advocated the requirement for research into botnet detection in cloud environments. The botnet threat, both to and from the cloud, is real, whilst hypervisor vulnerabilities expose the cloud infrastructure to attack. AV and VM introspection techniques have limitations in obtaining the profile of a botnet attack, where it is essential to identify the C&C servers in order to facilitate takedown of the botnet.

The communication systems that make botnets such a powerful attack force are also their Achilles heel. The regular chatter between bots and their C&C servers can be used for detection. In high-speed data networks, traffic capture methods such as PCAP, can result in volumes of big data. This big data requires subsequent analysis in order to locate a few botnet signature packets that could be only a few bytes in size. Not only is analysis at this scale difficult, but packet capture also results in huge quantities of accumulated storage data in a short period of time (*Hofstede, et al., 2014*). Flow protocols, such as NetFlow, are used by over 80% of network operators to capture network traffic management and reporting statistics (*Steinberger, et al., 2013*). Many researchers have repurposed NetFlow to capture network traffic for botnet detection. However, NetFlow was designed to be a network management protocol and presents limitations when applied to security threat analysis. In 2013, IPFIX (RFC-7011) was developed to overcome the weaknesses of NetFlow.

Chapter 1 identified a gap in knowledge which formulated the hypothesis for this research study; that the enhancements of standards-based IPFIX overcome the weaknesses of proprietary NetFlow, when applied to in botnet communication traffic capture. This chapter reviews the available literature to compare these two protocols, presenting an argument that IPFIX does exhibit advantages in a CSP environment that warrant further understanding. This chapter also reviews the literature that discusses the state of the art in botnet traffic capture using flow protocols.

3.2 A Brief History of Flow

In the 1980s SNMP (Simple Network Management Protocol) was the standard for network management. SNMP was designed to poll a proprietary MIB (Management Information Base) in a device every *x* number of minutes for basic information, such as up/down status and common error alerts, regardless of any change in the device state. However, the information obtainable from the MIB was limited. SNMP was not designed to carry a large amount of data and the protocol come with overheads. If more granular information was needed, Syslog would be used alongside SNMP, allowing devices to push information after an event or status change, rather than require regular polling. Syslog could be used for the efficient logging of device information, but an unstructured data format made it slow for querying and reports.

In 1991, the Internet Engineering Task Force (IETF) proposed aggregating packets into flows using packet header information for Internet accounting in RFC-1272 (*Internet Engineering Task Force, 1991*). The working group was disbanded in 1993 due to a lack of vendor interest. In 1996 a new working group was tasked with developing an architecture for flow measurement. A generic framework for Real-time Traffic Flow Management (RTFM) was published in 1999 as RFC-2721 (*Internet Engineering Task Force, 1999*). RTFM was a network flow metering process based on SNMP. Meanwhile Cisco was working on a proprietary flow export technology to speed up layer 3 packet switching called NetFlow. This precursor work became Cisco Express Forwarding - where forwarding decisions are made on the first packet of a flow, with subsequent packets being switched. Cisco patented NetFlow in 1996 (*Kerr and Bruins, 2001*). Cisco continued to develop NetFlow for network management. Multiple versions were developed, with the first commercial release being NetFlow v5 in 2002. In 2004 Cisco introduced NetFlow v9 (known as Flexible NetFlow, or FnF) which provided much needed improvements over NetFlow v5; including support for templates, VLANs, IPv6 and MPLS, amongst other features. Meanwhile, other vendors developed their own proprietary NetFlow protocols; none of which were interoperable.

In 2004 the IETF recognised the need for a standardised approach. In 2008 the first specifications of IPFIX were drawn up, based on NetFlow v9 as the underlying building block. In 2013 IPFIX was made the standard for flow protocol export under RFC-7011 through RFC-7015 (*Internet Engineering Task Force, 2013a; 2013b; 2013c; 2013d; 2013e*) and RFC-5103 (*Network Working Group, 2008*).

IPFIX was defined to provide an extensible, flexible data model that is reliable, secure and congestion-aware (*Trammell and Boschi, 2011*). IPFIX is a push protocol, where a device regularly sends IPFIX flow messages to collectors without the collector having to specifically request the data. Flow exports a highly structured dataset, where structured means that the data adheres to a pre-defined model that is organised in a predetermined way (*Santos, 2016*). When traffic capture in high-speed data networks produces big data, a structured dataset is an advantage not only for reporting, but also when it comes to analysis and querying; for example, for malicious network activity. IPFIX supports structured data as documented in RFC-6313 (*Internet Engineering Task Force, 2011).* As network operators are already capturing flow data for network management purposes, it makes sense to understand if these datasets can be re-purposed for other activities such as botnet detection.

3.3 Flow Export Architecture

Network monitoring is typically classified as either active or passive. Active monitoring, such as ping or traceroute, injects traffic into a network to perform measurements. Active monitoring has the potential to impact the traffic under monitor through introduced latency. Passive monitoring, such as PCAP, observes traffic as it passes a measurement point. Flow export falls into the passive monitoring category (*Hofstede, et al., 2014*). RFC-7011 defines flow as:

> "*… a set of packets or frames passing an Observation Point in the network during a certain time interval. All packets belonging to a particular flow have a common set of properties.*" (*Internet Engineering Task Force, 2013a*)

Figure 5, below, describes a flow-based monitoring architecture. A flow probe (or flow exporter), installed on the device being monitored, observes network traffic packets. This probe either reads packets directly from a monitored link via a network tap, or receives packets via the packet forwarding mechanism in the device being monitored. The probe then exports the flow records using the IPFIX or NetFlow protocols to a flow collector. The collector aggregates the flows based on a set of common properties, thereby reducing the overall amount of traffic collected for storage. Stored flow data is then automatically or manually analysed. *Hofstede, et al,* (2014) provide an excellent overview of the flow monitoring and export process.

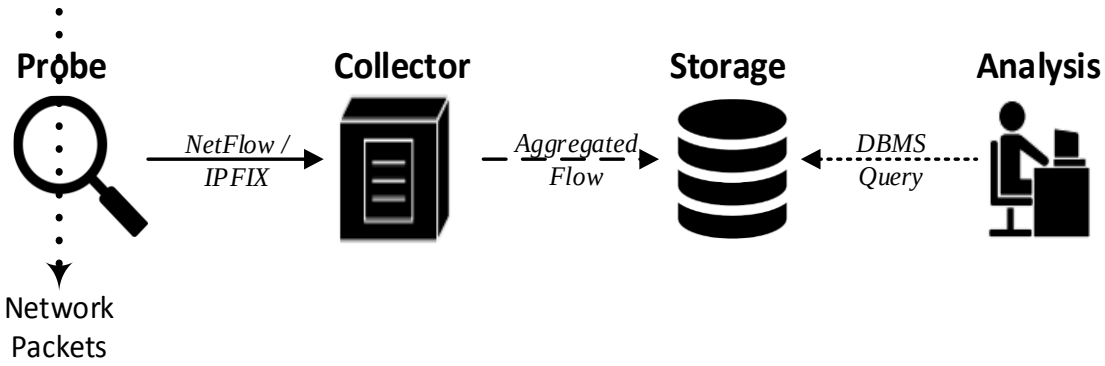


Figure 5. The four elements of a flow monitoring architecture.
(Adapted from Hofstede et al., 2014)

3.4 IPFIX Compared with NetFlow

Flow export protocols have evolved over time, primarily in order to overcome operational drawbacks. The application of NetFlow v5 in security analysis is restricted by its fixed template (*Gates et al., 2004*) which limits the number of traffic attributes that can be exported. This was overcome in NetFlow v9, which introduced customisable templates. However, the proprietary format of NetFlow v9 presented vendor interoperability issues, which drove the IETF to standardise flow export as IPFIX in 2013. NetFlow v9 is considered a protocol in its own right and is considered in the following section for purposes of clarity. However, as IPFIX was developed from Cisco's implementation of the NetFlow v9 protocol, throughout the remainder of this thesis NetFlow v9 is assumed to be an early, proprietary version of IPFIX and is generally ignored.

In addressing research objective #1, the following section outlines a comparison of IPFIX with NetFlow features that impact the design of a botnet traffic capture mechanism in high throughput network environments. Table 1 summarises how IPFIX morphed from a simple network management statistics protocol as NetFlow, into a protocol which application to security analysis. Table 1 shows how IPFIX evolved from NetFlow v5 to address template extensibility, the lack of security features in NetFlow, standardisation in data structure and transport protocols, and feature support for modern data networks such as multi-cast, VLANs and IPv6.

TABLE 1. FEATURE COMPARISON BETWEEN NETFLOW V5, NETFLOW V9 AND IPFIX PROTOCOLS (SUMMARISED FOR BOTNET TRAFFIC CAPTURE ATTRIBUTES)

	NetFlow v5	**NetFlow v9**[1]	**IPFIX**[2]
Commercial Release	2002	2004	2013
Standards Based	Proprietary	Proprietary	Standardised (RFC-7011)
Template Elements	18	79 (Cisco: 105)	386
Template Extensibility	Fixed Template	Yes	RFC-5610
Information Elements	No	Single-vendor ID	Multi-vendor ID RFC-7012, RFC-7013
Enterprise Elements	No	No	RFC-5610
Variable Length Fields	No	No	RFC-6313
Structured Data	Yes	Yes	RFC-6313
Transport Protocol	UDP	UDP (Cisco: UDP, SCTP)	TCP, UDP SCTP (RFC-6526)
Flow Security	None	None	Encryption, Integrity
Cache Timeouts	Fixed	Fixed	Customisable
IPv6	No	Cisco only	Yes
VLAN	No	Cisco only	Yes
MPLS	No	Cisco only	Yes
IPsec Tunnelling	No	Cisco only	Yes
MAC Address	No	Cisco only	Yes
Multi-cast	No	Cisco only	Yes
Flow Direction	Unidirectional	Unidirectional (Cisco: Biflow)	Bidirectional (RFC-5103)

[1] RFC-3954 (Informational) (*Network Working Group, 2004*)
[2] RFC-7011 to RFC 7015 (*Internet Engineering Task Force, 2013a; 2013b; 2013c; 2013d; 2013e)*

3.4.1 VENDOR INTEROPERABILITY

Arguably the biggest drawback of NetFlow comes from a lack of standardisation across vendor implementations of the protocol. Most manufacturers of network hardware have their own proprietary version of a flow export protocol; Cisco's NetFlow, Juniper's JFlow, Alcatel-Lucent's CFlow, Huawei's NetStream, Citrix's AppFlow and so on. Any moderately sized organisation is unlikely to deploy just a single vendor's equipment throughout their network, particularly if they desire a best-of-breed implementation. A typical multi-tenant CSP infrastructure uses servers, physical virtualisation, network virtualisation and LAN devices, data storage and management requirements such as accounting and security. No one single vendor addresses this entire hardware and software remit. Each has their own flow implementation. This does not present such an issue for NetFlow v5, which has relatively limited functionality. However, when NetFlow v9 proprietary features are deployed, interoperability becomes an issue. This may explain why NetFlow v5 is still the most common flow export protocol. In constructing a botnet traffic capture mechanism non-interoperability presents two issues. One, the highly structured format of flow data becomes lost as different vendors implement their own features using various methods, thus making querying and analysis of captured data more complex. Two, there is no guarantee that all vendors export the same traffic attributes, thus resulting in blind spots across a distributed network. The standardisation of the flow protocol to IPFIX addresses vendor non-interoperability issues, as is evidenced by the features investigated below.

3.4.2 TEMPLATE EXTENSIBILITY

NetFlow v5 exports a fixed structure of 20 fields; comprising 18 data fields with two padding fields. Figure 15, below, shows the NetFlow v5 template structure. These 20 fields are pre-defined and cannot be changed, removed or added to, regardless of whether the information needs capturing or not. This means that each NetFlow v5 packet is always 48 bytes in length. Gates argued that only 10 fields provide useful data when NetFlow is applied to security analysis (*Gates et al., 2004*). Thereby making the fixed template structure of NetFlow v5 a drawback, as effectively 18 bytes of each 48 byte NetFlow v5 packet is superfluous when used in security analysis. NetFlow v9 introduced customisable templates, allowing a flow device to define the information to be exported. A NetFlow v9 template can contain any quantity of fields and is no longer capped at 18 fields. Each NetFlow v9 template is identifiable by its

template ID, which corresponds to the set ID in the set header section of the dataset. NetFlow v9 is documented in RFC-3954 (*Network Working Group, 2004*), which outlines 79 different fields that are supported in NetFlow v9 templates. However, NetFlow v9 has not been standardised, which has led to differing vendor implementation of NetFlow v9 arising. For example, Cisco's implementation of NetFlow v9 offers 105 fields, with fields 106-127 reserved for future use (*Santos, 2016*). There is no guarantee that the 79 defined fields, or Cisco's additional fields, will operate between vendors implementation. Many of the NetFlow v9 fields focus on capturing flow contextual information, such as byte or packet counters, rather than traffic content information.

IPFIX overcomes field interoperability by standardising the template format and template fields in RFC-7012 (*Internet Engineering Task Force, 2013b*). With IPFIX, template fields are called Information Elements (IEs). The RFC does not define the IEs themselves. Instead, the RFC states that in order to maintain cross-vendor interoperability, the Internet Assigned Numbers Authority (IANA) is responsible for controlling the available IPFIX IEs. IANA defines 433 standard fields in the IPFIX Information Element Registry[3]. Of these 433 IEs, 17 are now deprecated, with another 57 IEs preserved for NetFlow v9 fields so as to retain compatibility between NetFlow v9 and IPFIX. IPFIX was designed as an extensible data model for flexibility and customisation. IPFIX supports the creation of new template fields if they are not defined in the 433 IANA fields. With IPFIX, new template fields are called Enterprise Elements (EEs). The IPFIX IEs range 434 to 32767 is reserved for vendor defined EEs. Effectively, EEs allow the addition of any Layer 2 through Layer 7 information to the IPFIX template. This means that as new threat detection techniques are developed, new fields can be created to be added to the IPFIX template. *Trammell and Boschi* (2011) state that EEs allow IPFIX to extend flow collection beyond network and transport layers, making it able to export information from future networks. Whilst IPFIX template extension support is documented in RFC-5610 (*Network Working Group, 2009b*), currently only a few IPFIX vendors support creation of new EEs. Where EEs are supported, this is usually done through open source libraries such as libpcap. A list of IPFIX probes that support EEs is outlined in Chapter 5. IPFIX also supports Option Templates. Whilst an IPFIX template describe the capture flow, option templates define non-flow attributes such as flow metadata, collection infrastructure or other properties of a set of flows (*Santos, 2016*). NetFlow v9 does not support vendor extensions to allow capture of new data attributes (*Patterson, 2012*).

[3] https://www.iana.org/assignments/ipfix/ipfix.xhtml

NetFlow v5 lacks support to capture network information such as IPv6, VLANs, MPLS, IPsec tunnelling, MAC addresses or multi-cast. The fixed nature of the NetFlow v5 template structure prevents support for these features being added. As cloud providers utilise these network functions, it is essential for a data capture mechanism to support these. Likewise, with the IPv4 address range now exhausted, the IoT will take advantage of the massive address space in IPv6. IPFIX supports all of these features as IEs, whilst EEs allow the creation of new template attributes to accommodate new technology and next-generation protocols. NetFlow v9 lacks support for these features, apart from Cisco's implementation. When studying anomalous IPv6 traffic *Lee, et al.,* (2007) chose IPFIX over NetFlow v9 because of the ease in which new EEs could be constructed to capture IPv6 attributes. In a botnet traffic capture mechanism, as bots evolve with new features, EEs can be created so that new detection attributes can be included into existing IPFIX capture templates. In particular this EE space can be used for protocol specific attributes, such as HTTP GET information that may allow confirmation that suspect traffic is indeed malicious. *Velan* (2013) argues that the lack template customisation makes both NetFlow v5 and NetFlow v9 limited for threat detection, as they only analyse a packet's encapsulation protocol rather than the packet itself. *Velan, Jirsik and Čeleda* (2013) go on to state that support for EEs means IPFIX will be superior to NetFlow in next-generation network monitoring, whilst supporting higher collection performance and better use in analysis tools.

3.4.3 VARIABLE LENGTH FIELDS

NetFlow not only lacks the functionality to create new template elements, neither does it support variable length fields. A NetFlow v5 fixed 18 field template is always 48 bytes in length, regardless of the data captured. Whilst fields can be added and removed to the NetFlow v9 template, attribute capture is limited to fixed length fields. Variable length fields are required to efficiently capture variable length strings. In a botnet traffic capture mechanism this allows the creation of EEs to capture strings such as HTTP GET requests, SMTP Hellos, IRC messages, or any other Layer 2 through Layer 7 attributes which would help in facilitating confirmation that suspect traffic is indeed malicious. Most Layer 7 attributes tend to be variable length strings. IPFIX supports variable length fields as documented in RFC-6313 (*Internet Engineering Task Force, 2011).* Furthermore, it achieves this whilst continuing to maintain the data structure, allowing variable length fields to be used without any impact upon post-capture data analysis. The method in which IPFIX supports variable

length fields and structured data prompted *Hofstede, et al.,* (2014) to state that IPFIX should not only be considered as the flow export protocol of choice, but also the generic transport protocol for structured data. It may be possible for NetFlow v9 to support variable length fields by creating out-sized fields and using field padding. Although this approach wastes space within captured packets. Additionally, NetFlow v9 does not support the creation of EEs, making this argument arbitrary.

3.4.4 TRANSPORT PROTOCOL

When early NetFlow protocols were designed it was expected that flow records would be confined to private networks, with flow exporters and flow collectors in close proximity to each other. Both NetFlow v5 and NetFlow v9 use UDP as the transport protocol. In a large network, using UDP as a transport layer protocol introduces two limitations. UDP lacks any congestion awareness, which can leading to network flooding when a device is down or undergoing a DDoS attack. UDP is an unreliable protocol lacking data re-transmission, making it susceptible to data loss. A design requirement of IPFIX was to address these reliability and congestion-awareness issues, whilst remaining transport protocol independent (*Trammell and Boschi, 2011*). The transport protocol for IPFIX can be selected from UDP, TCP or SCTP (Stream Control Transmission Protocol). SCTP for IPFIX is documented in RFC-6526 (*Internet Engineering Task Force, 2012*). UDP is not recommended and is provided primarily to allow migration from NetFlow to IPFIX installations. Instead, SCTP is the recommended transport protocol for IPFIX as it addresses congestion awareness (*Santos, 2016*), which allows graceful degradation through selective dropping of exported datagrams under high load, rather than overloading buffers (*Internet Engineering Task Force, 2013a*). SCTP ensures the reliable transmission of IPFIX templates, thereby improving end-to-end delay whilst reducing dropped packet count and packet retransmissions. Whilst NetFlow does not support transport over TCP, Cisco's version of NetFlow v9 does support congestion awareness through NetFlow Reliable Export with SCTP.

3.4.5 FLOW SECURITY

An early draft of IPFIX highlighted the lack of security in NetFlow (*Network Working Group, 2004*). Confidentiality, integrity and authentication were not implemented in NetFlow as they reduced the efficiency of the protocol. This makes NetFlow vulnerable to many forms of attack, such as man-in-the-middle (MITM) attacks,

packet tampering, packet forgery and attacks upon the collector. Security requirements for IPFIX are documented in RFC-7011 (*Internet Engineering Task Force, 2013a*). This mandates that IPFIX transmission includes authentication to prevent MITM attacks, integrity to prevent IPFIX flow manipulation or duplication, and obfuscation for flow record confidentiality. RFC-7011 recommends that when TCP is used as the transport protocol, transmission should use TLS (Transport Layer Security) v1.2 or above and SYN cookies are used for protection against DDoS attacks. However, SCTP can be difficult to transmit over the Internet as some devices will drop SCTP packets due to unrecognised protocol numbers (*Hofstede, et al., 2014*). To overcome this, transmission of SCTP over the Internet is recommended with TLS. A drawback to this is that it requires bidirectional streams with one TLS connection per stream, thereby contradicting the reasons for selecting SCTP as an IPFIX transport protocol in the first place. Therefore, RFC-7011 recommends that when using SCTP, or UDP, as a transport protocol, that DTLS (Datagram Transport Layer Security) is the preferred security mechanism. When using DTLS, it is essential that IPFIX messaging is sent over the same SCTP stream to prevent injection attacks. When using SCTP, RFC-7011 mandates a cookie exchange mechanism. RFC-7011 recommends TLS or DTLS is used to prevent fake IPFIX messages when transportation uses SCTP or UDP. SCTP does not mandate encryption however several IPFIX probes implement flow encryption by using TLS (see Chapter 5 for more information). In a botnet traffic capture mechanism within a network where the detection system itself is at risk, such as a CSP or IoT environment, IPFIX provides mitigation of attacks such as DDoS, MITM and packet manipulation.

3.4.6 CONFIGURABLE CACHES TIMEOUTS

Flow exporters maintain a flow cache table in the device memory to track all known active flows passing through the device. Flows are cached until either the timeout settings hit a threshold, or the flow terminates; for example through a TCP FIN or RST flag. In NetFlow the timeout setting is fixed according to the device flow cache size, which in turn is constrained by the device hardware or software. This is usually set to 60 seconds (*Patterson, 2012*). If malware can generate enough traffic, for example via a DDoS attack, with a high flow cache timeout setting it could be possible to exceed the maximum number of permissible flow cache entries. Thus forcing the flow exporter to drop flows, or crash. Similarly, because the number of flow records coming into a collector can escalate considerably during a DDoS attack, if cache timeouts are high enough, the flow exporter can overload its collector, forcing

the collector to error or crash (*Hofstede, et al., 2014*). IPFIX neither mandates a time period after which flow entries expire, or enforces a duration after which flow records are forced to be exported. This has two advantages for a botnet traffic capture mechanism. It protects the flow capture infrastructure from DDoS attack. It also allows optimisation of flow expiry and idle time so IPFIX can capture short-lived communication bursts, such as a bot to C&C server keep-alive beacons.

3.4.7 BIDIRECTIONAL FLOWS

Most networked host-to-host communications involve packet exchanges in both directions. For example, TCP is two directional since it relies on packet acknowledgement. NetFlow v5 is unidirectional, hence TCP is captured as two non-interconnected flows; namely a request and response. IPFIX provides support for bidirectional flows (biflow) as documented in RFC-5103 (*Network Working Group, 2008*). Many IPFIX exporters claim to support biflow, but in reality they recognise flow pairs through record adjacency. It is possible to pair flows together in NetFlow v5 and NetFlow v9, but this requires the creation of an algorithm to undertaken flow pairing during analysis (*Minarik, Vykopal and Krmicek, 2009*). Only Cisco's implementation of NetFlow v9 supports biflow. A biflow is a single record which contains both the traffic details from A to B and B back to A (*Patterson, 2012*). This allows request and responses to be individually distinguished and interconnected as a biflow pair. Biflow pairing is becoming more important in security analysis by associating inbound and outbound flows to application. For example, biflow pairing determines which party initiated the conversation, particularly useful for P2P traffic study. *Yen and Reiter* (2010) used biflow ratios to detect the Storm P2P botnet. Studies in packet symmetry have shown that the ratio of inbound to outbound packets can be useful in determining malicious traffic (*Kreibich, et al., 2005; Lee and Brownlee, 2007*). For example, separating the request and reply pairs of server, client and single flows can recognise a distributed attack against a DNS server (*Minarik, Vykopal and Krmicek, 2009*). In a botnet traffic capture mechanism biflow pairs would allow the determination of the direction of conversation between a C&C server and a victim. Separation of unanswered from answered TCP requests may suggest a C&C server searching for a botnet peer that is offline, or a C&C server undertaking a scan for victims. Alternatively, rather than use Biflow, flow pairs can be distinguished by collecting both source and destination IP addresses, which can be a more efficient method of capture when used with data aggregation (*Patterson, 2012*).

3.5 Flow Export Compared with Packet Capture

Flow export and packet capture are both *passive* monitoring techniques, observing traffic as it passes a measurement point and making a copy of some or all of this traffic. PCAP captures both packet header and payload information. This makes PCAP capture a big data challenge in high-speed data networks as it can capture gigabytes of data per second. Flow export captures only information in the packet header, ignoring the payload data. Flow export therefore captures the same metadata as PCAP, but the overall amount of traffic capture for storage is considerably less without the payload. In network monitoring, flow export tends to be used for high level investigation within a network, with PCAP utilised for deep dive investigation requiring payload information.

Volumes of exported flow data can be further reduced by real-time data aggregation. Aggregation capability is typically a feature of the exporter/collector pair, rather than defined by the flow protocol. IPFIX however standardises aggregation, as detailed in RFC-7015 (*Internet Engineering Task Force, 2013e*). RFC-7015 defines aggregated flows as "flows representing packets from multiple original flows sharing some set of common properties". Similar flows are aggregated by key field tuples such as {source IP, destination IP, source port, protocol}. Should 10 flows match this tuple over a fixed collection period, they are exported as one aggregated flow. Other traffic monitoring features, such as Cisco's SPAN, do not support aggregation by default.

From a CSP perspective, flow export has advantages over packet capture:

- Flow export allows data storage volume savings; in the order of $1/2000^{th}$ of the original PCAP volume (*Hofstede, et al., 2014*);
- Flow export is less privacy sensitive (*Hofstede, et al., 2014*); since only the packet headers are considered. *Note: IPFIX EEs can be constructed to capture payload data*;
- Flow export data complies with European data retention laws (*Hofstede, et al., 2014*) where European service providers are legally obliged to retain connection data from between six months to two years for the purpose of "prevention, investigation, detection and prosecution of criminal offences" (*EC Data Retention Directive 2006/24/EC, 2006*);

- Flow export is more suitable for high-speed data networks (*Hofstede, et al., 2014*) as not capturing packet payloads make it more scalable and able to cope with high-speed data network throughput over 100Gbps where packet capture is traditionally limited or requires expensive hardware.

3.5.1 A NOTE ON SFLOW AND OPENFLOW

NetFlow and IPFIX are flow *export* protocols. Other protocols exist with "flow" in their name, but these are not flow export protocols, so are beyond the scope of this research project:

- sFlow is a packet sampling protocol, capturing every 1:X packets. Flow export, by default, captures every packet. In large networks where export rates become prohibitively high due to network throughput, it is more common to see NetFlow revert to 1:X sampling to reduce data volumes and processor overheads on NetFlow devices.

  The drawback of sampling is that only capturing every X^{th} packet may miss short-lived inter-botnet communications. Caching can be another drawback of sFlow. With flow export the flow cache resides on the export device. With sFlow it is common to see the flow cache external to the sFlow device hence aggregation is not possible. If necessary, IPFIX supports packet sampling (PSAMP) as defined in RFC-5476 (*Network Working Group, 2009a*);

- OpenFlow is a software-defined packet forwarding protocol. OpenFlow is used to transport the routing decisions between a logically centralised controller and the data plane of a software switch (*Internet Research Task Force, 2015*).

3.6 Flow-Based Botnet Detection

Total eradication of a botnet requires the takedown of all C&C servers associated with that bot. Signature-based techniques such as AV software can disinfect individual machines to remove part of the botnet, but is not capable of locating and removing the C&C servers, as outlined above in Chapter 2. Signature-based detection is also heavily reliant upon analysing the packet payload content, which is both resource intensive and can be evaded by payload encryption (*Zhao, et al., 2013*). In a CSP environment, packet inspection also raises considerations around tenant privacy.

Botnet behaviour-based detection exploits uniformities in both botnet communication and behaviour, such as the communication between a recruited victim and its C&C server (*Gu, Zhang and Lee, 2008*). In behaviour-based detection, network traffic is fed into a detection engine. Typically, the detection engine comprises two stages. First, the incoming data is filtered, such as clustering or correlation, to reduce the data volumes requiring analysis. Second, this filtered data is then fed into a detection algorithm, such as decision tree or a machine learning algorithm. Behaviour-based detection has two disadvantages:

(1) Detection algorithms tend to be formulated based upon which attributes can be captured. If the data capture mechanism is based on NetFlow v5, detection attributes are limited to 18, increasing to 79 when NetFlow v9 is used. If the attributes are unable to be captured in NetFlow, capture is subsidised with PCAP. Although the additional volumes of data captured by PCAP can hinder analysis. However, *Sperotto, et al.,* (2010) considered packet inspection as complementary to flow-based capture techniques, where a combination of both NetFlow and PCAP may improve detection accuracy. This is at a cost of having to correlate multiple data feeds of varying structures.

(2) Because a data feed is a stream of traffic, it can take time to build up a traffic profile of the botnet. Therefore it is important that the detection algorithm is able to detect a bot as early in the bot life cycle as possible. However, this also provides an advantage. If a detection algorithm works at detection early in the life cycle, before the botnet attacks, a window of time exists before mitigation action needs to be taken. This window provides time to confirm that the suspect traffic is indeed botnet traffic. In other words, detection only needs to happen in near real-time.

BOTNET DETECTION EXPERIMENTS

The remainder of this section reviews the prior art in botnet detection using flow protocols, in chronological order. To address research objective #2 and the conceptual development of an IPFIX capture template, traffic attributes used by previous researchers must be understood. Therefore, this review of prior studies concentrates on data collection methodologies and the attributes captured, rather than the detection algorithms themselves. What was apparent from the literature review was that whilst some traffic attributes are more popular than others (refer to Table 2), very few authors provide evidence that their chosen attributes are empirically justified as botnet traffic indicators. Chapter 4 addresses this by considering the frequency and duplicity of all available IPFIX IEs and EEs. Table 2, below, provides a summary of each NetFlow and non-NetFlow attribute collected by each researcher as data feeds into both C&C and P2P detection algorithms. In Table 2 in can be seen how almost all prior studies have used NetFlow v5, packet capture, or a combination of the two. These is a core set of about ten NetFlow v5 attributes that are captured across all studies, whilst other NetFlow v5 attributes are more rarely used. Indeed, the ***nextHop*** NetFlow v5 field is not used by any detection algorithms. A traffic capture engine based on NetFlow v5 captures all 18 data field regardless of their use, resulting in redundant data that must be removed before analysis. More recent studies have begun to use NetFlow v9 (*Wijesinghe, Tupakula and Varadharajan, 2015; Haddadi, et al., 2014*). These both added flow contextual fields to the traditional NetFlow v5 template, yet failed to take advantage of other fields available in NetFlow v9. No bot detection studies can be found that take advantage of the power of IPFIX in data capture.

Some of the earliest research into the use of flow protocols in network security was performed by *Gates, et al.,* (2004). From studies of the Korgo and Sasser worms, they propose that eight fields in NetFlow v5 are superfluous for malware detection; namely input interface, output interface, source AS, destination AS, source mask, destination mask, next hop IP and type of service. However, as NetFlow v5 has a fixed template, these fields cannot be removed so are captured regardless of their value to threat detection. Hence, when NetFlow v5 is used in threat detection its inefficiency results in data volume and storage wastage. *Cooke, Jahanian and McPherson*, (2005) found that introducing proxy servers as a stealth layer in IRC botnets prevented detection of distinguishing C&C server traffic characteristics. They conclude that payload inspection is time and resource costly and subject to encryption. They instead recommended NetFlow be used to search for non-humanlike traffic characteristics of bot attack/propagation traffic. BLINC (*Karagiannis, Papagiannaki and Faloutsos,*

2005) was created to distinguish between benign and malicious P2P traffic by classifying traffic patterns at the transport layer. They found that P2P flows with no payloads were indicators of port scanning or IP address scanning. As BLINC only uses NetFlow for data capture, it was unable to characterise specific P2P protocols as this requires data found in the payload. They decline the use of port numbers in detection due to possibilities for port spoofing. Rishi (*Goebel and Holz, 2007*) use regular expression (regex) searches on traffic connection time, source/destination IP address, source/destination port, IRC channel and IRC nickname to extract malicious IRC channels. They chose to use packet capture because NetFlow v5 does not support IRC attributes. This experiment could be reproduced using IPFIX EEs to extract IRC data. *Karasaridis, Rexroad and Hoeflin* (2007) captured mainly transport-layer attributes to detect IRC botnets in tier-1 ISPs. They found that idle IRC clients produced different traffic patterns to active IRC clients. They preferred NetFlow as it was non-intrusive, respected privacy and generated considerably less traffic than PCAP. However, to be able to confirm their detection engine was capturing IRC bots, they used packet capture to extract application layer data.

In the first of three related studies, BotHunter (*Gu, et al., 2007*) claimed to be the "first distributed bot infection profile analysis tool." BotHunter analysed payload data from SNORT IDS running a customised malware ruleset, to look for IRC bot scanning, infection and keep-alive communication patterns. BotSniffer (*Gu, Zhang and Lee, 2008*) used the same SNORT IDS to analyse payload data, but improved the BotHunter correlation engine to detect IRC bots, HTTP bots and look to for SPAM by correlating DNS with SMTP traffic. Two anomaly detection algorithms looked for high-scan rates and high failed connection rates based on the spatial-temporal correlation of network traffic worked under the premise that bots have much stronger and more consistent synchronisation and correlation in their responses compared to human users. They suggested that looking for user-initiated IRC queries, such as WHOIS, LIST and NAMES, could indicate benign traffic, as malicious traffic is unlikely to use these commands. BotMiner (*Gu, et al., 2008*) again enhanced the BotHunter platform to create an engine for high-speed, low packet loss networks, to detect IRC, HTTP and P2P bots that does not require priori knowledge of bot signatures. A proprietary NetFlow format is used to extract transport layer traffic, whilst packet capture is used to capture for application layer information such as SMTP and DNS record attributes. BotHunter's weakness was that in looking for pre-defined bot life cycle patterns, it could not detect a bot if its infection model changed.

Strayer, et al., (2008) feed flow characteristics into machine learning correlation

algorithms to pro-actively look for IRC botnet hosts. The algorithm uses discriminatory flow attributes such as flow duration, flow direction, average bytes-per-packet per flow, average packets-per-second per flow and average bits-per-second flow. Unfortunately, some classification algorithms presented false positives of up to 40%, rendering this technique questionable. Also questionable is the amount of potentially useful detection data that is filtered out at an early stage without evidence. This includes port scanning, high bandwidth flows and short lived flows. Botlab (*John, et al., 2009*) looks for spam-bots, again using a customised flow extracted from packet capture. They create application-layer behavioural signatures in order to attribute incoming spam to a specific bot. Botlab was successful because the bots analysed used either hardcoded IP addresses or DNS information to locate their C&C servers. The fluxing technique used by more recent botnets may make these signatures more difficult to create. *Wurzinger, et al.,* (2009) hypothesises that when a bot receives commands from its botmaster, each bot must respond in its own specific way. They use IDS to extract payload traffic using packet capture, to generate bot signatures based on network traffic, command/response pairs and IRC and HTTP application traffic. These signatures are then compared against pre-generated models for IRC, HTTP and P2P bots. They acknowledge that a drawback to this technique is that it requires pre-generated signature models before it can detect a particular bot.

BotGrep (*Nagaraja, et al., 2010*) uses NetFlow v5 to capture traffic on high-speed ISP networks, then applies Graph theory to detect P2P bot traces. To be able to cope with throughput at high-speed, they use NetFlow to sample network traffic at 1:500 rates. A limitation to sampling in that it can miss the few tell-tale botnet packets, so detection accuracy will suffer. Also, they fail to state the NetFlow attributes captured. *Perdisci, Lee and Feamster* (2010) analyse the structural similarities among malicious HTTP malware to create signatures. They focus on HTTP malware rather than bots. However, their HTTP attributes could apply to HTTP botnet detection. They split HTTP attributes into (1) course-grain - statistical values of HTTP request, GET/POST, URL lengths, number of request parameters and (2) fine-grain - clustering by HTTP structural similarity of HTTP GET/POST and URL.

To detect P2P traffic, BotTrack (*Françios, Wang and Engel, 2011*) uses NetFlow to capture only IP addresses, which are clustered into dependency graphs and analysed using a PageRank algorithm. They prefer NetFlow to packet capture due to NetFlow's collection efficiency, whilst non-packet inspection characteristics address privacy concerns. In BotCloud, *Françios, et al.,* (2011) replace the resource-intensive PageRank component of BotTrack with MapReduce and a mini Hadoop cluster to

reduce computational time by a factor of seven. In order to improve capture efficiency both BotTrack and BotCloud capture a limited set of NetFlow attributes, without any justification for ignoring other bot attributes that may enhance the detection algorithm. *Rossow, et al.,* (2011) use a Sandnet environment to analyse the network behaviour of malicious software that uses HTTP and DNS protocols. The value of their work is in identifying common malware HTTP and DNS attributes from analysing 70 million flows from 207GB of data. Packet capture was used to capture this data, but IPFIX EEs could enable collection of the same attributes at a fraction of this space. DISCLOSURE (*Bilge, et al., 2012*) searches for C&C servers within high-speed tier-1 ISP networks, extracting flows with different incoming/outgoing characteristics as potential bots. They found that the size of flow from benign servers fluctuate measurably more than from C&C servers. They selected NetFlow v5 for data capture because it is commonly used by ISPs, but found some limitations of NetFlow. It does not capture payload data which is required for bot detection. It is unidirectional so only captures one side of the conversation. It struggles in aggressive sampling rates required for DISCLOSURE to work in high-speed ISP networks. All of these issues can be addressed by IPFIX. *Zhang, et al.,* (2011) also had issues with data volumes in packet capture sampling in high-speed ISP networks. They developed a sampling mechanism for PCAP, which reassembles flows based on aggregating key fields. This methodology essentially provides a NetFlow format that can capture packet payload data, although this is computationally costly. IPFIX could capture the same data more efficiently. Additionally, if network speeds become too high for IPFIX to efficiently aggregate the traffic, IPFIX supports PSAMP for sampling. This system has since been enhanced, using NetFlow and packet capture, to statistically fingerprint P2P traffic to determine if it is belongs to a legitimate P2P network (*Zhang, et al., 2014*). *Zhao, et al.,* (2013) split complete flows into multiple shorter time windows to improve overall detection speed, enabling detection of botnet behavioural patterns early in the propagation life cycle. Packet capture was used to capture flow contextual data to show bot P2P communication, which displays many continuous, uniform, smaller sized packets, unlike benign P2P usage. To counter bot detection evasion mechanisms, such as packet injection or random reconnection, they measure P2P reconnects against total flows over time. NetFlow cannot capture this additional contextual information, however it is possible using IPFIX.

Yen and Reiter (2010) find that bot P2P traffic demonstrate boths lower peer churn with more failed connections and lower traffic volumes over shorter time-periods, than compared to human P2P file-sharing. Building upon this *Narang, Reddy and*

Hota (2013) compare three machine learning techniques to improve P2P bot detection speeds in IDS. Detection attributes are extracted from PCAP and aggregated into custom flows, before being input into their machine learning detection algorithms. They find that reducing the number of attributes in a flow, such as port numbers and protocols both of which can be spoofed, increases the detection speed with only a marginal loss of accuracy. To detect P2P botnets in their quieter period before attack *Hang, Wei and Faloutsos* (2013) proposed superflows as a technique to improve the low accuracy of Yen and Reiter's work. Where superflows are flows with the same IP nodes that are close in time, irrespective of protocol or port number. They claim that superflows should be able to overcome the proprietary nature of flow export tools. Similarly, PeerShark (*Narang, et al., 2014*) uses packet capture to create bespoke conversations based on flow length and duration. They find that conversations can detect the pre-attack stealthy, low-volume conversations between botmaster and modern P2P bots. Again, these experiments can all be reproduced using IPFIX.

Based upon observations that malware authors use various encoding schemes, such as Base64, Hex or ASCII, to obfuscate HTTP-based C&C channels, CoCoSpot (*Dietrich, Rossow and Pohlmann, 2013*) uses the length of the first 8 messages of a flow, protocol and URI encoding scheme to detect HTTP botnets. *Lin, Chen and Chang* (2014) use packet size and packet count from PCAP packet capture to classify malicious P2P bots against benign P2P flows. They find that P2P bots use more packets per session and that these packets, ranging from 63-399 bytes, are smaller than benign P2P traffic. *Haddadi, et al.,* (2014) capture HTTP flows using a NetFlow v9 template, before feeding them into a machine learning algorithm. Whilst they use NetFlow v9, the template is nothing more than NetFlow v5 with the addition of VLAN fields and some flow contextual fields. They also collect NetFlow input/output interfaces, AS addresses and IP masks, despite *Gates et al.* (2014) advice to the contrary. They indicate that their approach is not successful. This may be because they do not capture NetFlow fields such as IP addresses, port numbers and non-numeric data. *Wijesinghe, Tupakula and Varadharajan* (2015) claim to use an IPFIX template for traffic capture. However, their published results are from NetFlow v9, not IPFIX as claimed. Furthermore, their NetFlow v9 template is simply NetFlow v5 with the addition of a payload length field. As with Haddadi et al., they omit to take full advantage of the additional fields that NetFlow v9 can bring to botnet detection. *Garg, Peddoju and Sarje* (2016) found that different P2P protocols generate different failed connection profiles, but when coupled with new peer discovery by ports, P2P bots generate different profiles to benign P2P traffic.

TABLE 2. A SUMMARY OF BOTNET DETECTION EXPERIMENTS
LISTING THE INPUT ATTRIBUTES INTO THEIR DETECTION ALGORITHM, SINCE GATES 2004

				NFv5 Attributes																Non-NFv5 Attributes										
	Year	IRC HTTP P2P	P=Packet Capture 5=NFv5 9=NFv9	srcAddr	dstAddr	nextHop	input	output	dPkts	dOctets	first	last	srcPort	dstPort	tcpFlags	proto	tos	srcAS	dstAS	srcMask	dstMask	flowsTotal	payloadSize	srcVLAN	dstVLAN	ircHeader	ircChannel	httpURL	httpUserAgent	httperver	smtp	dns	bgpPrefix
Gates, et al.	2004	-	5	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓																	
BLINC Karagiannis,et al.	2005	P	5	✓	✓				✓	✓	✓	✓	✓	✓		✓						✓											
RISHI Goebel & Holtz	2007	I	P	✓	✓					✓	✓	✓	✓												✓	✓						
BOTHUNTER Gu, et al.	2007	I	P	✓	✓					✓	✓	✓	✓	✓	✓	✓									✓							
Karasaridis, et al.	2007	I	5	✓	✓				✓	✓	✓	✓	✓	✓	✓							✓			✓							
BOTMINER Gu, et al.	2008	I,H,P	5, P	✓	✓				✓	✓	✓	✓	✓	✓																✓	✓	
BOTSNIFFER Gu, et al.	2008	I, H	P	✓	✓					✓	✓	✓	✓												✓		✓			✓	✓	
Strayer, et al.	2008	I	P						✓	✓	✓	✓			✓	✓																
BOTLAB John, et al.	2009	H	P	✓	✓								✓	✓		✓														✓	✓	
Wurzinger, et al.	2009	I,H,P	P	✓	✓				✓	✓	✓	✓	✓	✓		✓							✓			✓	✓	✓	✓		✓	✓	
Perdisci, et al.	2010	H	P	✓	✓						✓	✓											✓					✓		✓		
Yen & Reiter	2010	P	5, P	✓	✓				✓	✓	✓	✓	✓	✓		✓												✓				
BOTRACK Francios et al.	2011	P	5	✓	✓						✓	✓																				
SANDNET Rossow, et al.	2011	H	5,P	✓	✓				✓				✓	✓		✓												✓	✓	✓	✓	✓	
Zhang, et al.	2011	I,H,P	P	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓																
DISCLOSURE Bilge, et al.	2012	H	5	✓	✓				✓	✓	✓	✓	✓	✓																		
COCOSPOT Dietrich, et al.	2013	H	P	✓	✓						✓	✓		✓								✓	✓					✓				
ENTELECHEIA Hang, et al.	2013	P	5	✓	✓						✓	✓	✓	✓		✓																
Narang et al.	2013	P	P	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓																
Zhao, et al.	2013	P	P	✓	✓				✓		✓	✓	✓	✓		✓						✓	✓										
Haddadi, et al.	2014	H	9				✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓			✓	✓							
Lin et al.	2014	P	P						✓						✓																	
PEERSHARK Narang, et al.	2014	P	P	✓	✓				✓	✓	✓	✓			✓								✓									
Zhang, et al.	2014	P	5,P	✓	✓				✓	✓	✓	✓			✓	✓															✓	✓
Wijesinghe, et al.	2015	I,H,P	9	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓							✓									
Garg, et al.	2016	P	P	✓	✓				✓		✓	✓	✓	✓	✓	✓																

3.7 Summary

In high-speed data networks such as a CSP environment, PCAP packet capture presents two drawbacks. Both of which arise because PCAP captures the packet payload as well as the packet header. PCAP raises a privacy debate, as it captures packet payloads which store content. This payload capture, in turn, makes PCAP a big data challenge, capturing Gigabytes of information every second. All of this data must be stored and later analysed.

Flow export is a more efficient packet capture method, allowing a finer granularity in data capture. Almost all research involving flow export protocols in botnet detection have utilised NetFlow to capture traffic attributes to feed into botnet detection algorithms. Where a specific bot attribute is not available for capture in the fixed NetFlow v5 template, flow export is used in conjunction with PCAP to capture missing attributes (*Sperotto, et al., 2010*). This approach has led to the development of many successful botnet detection algorithms. However, when multiple data feeds are used they must be correlated before analysis, as well as introducing high data storage demands, particularly when supplemented with PCAP data.

Chapter 1 articulated the gap in the knowledge in the understanding of how the next-generation of flow export protocols, such as IPFIX, can be applied to efficient botnet detection. Chapter 2 outlined failures in signature-based detection methods, such as AV, to track the botnet C&C servers; an essential step in botnet takedown. This chapter further contributes to addressing research objective #1 through a critical investigation into the suitability of IPFIX for botnet traffic capture, in a CSP. By taking a critical look at how IPFIX compares to NetFlow, evidence in presented in agreement with the hypothesis that IPFIX is more suited to threat detection than NetFlow. Table 1 summarises this critical review from the point of view of a botnet traffic capture mechanism for use in CSPs. The key features in which IPFIX addresses design weaknesses of NetFlow are through standards-based vender interoperability, a high degree of template extensibility, inherent protocol security, and support of modern network requirements such as IPv6 and VLANs. In particular, IPFIX template extensibility allows not only bespoke template creation, but also supports variable length fields which allow the creation of bespoke enterprise elements to capture data attributes such as variable length HTTP GET strings. These features will be applied to template construction in Chapter 4. The other key features of IPFIX will be used in Chapter 5, in the design of a botnet capture prototype.

Gates, et al., (2004) opined that the main weakness of NetFlow v5 in security analysis is it that it captures a fixed dataset that is not used in its entirety. This is echoed in Table 2 which shows that experiments which capture network traffic using NetFlow v5, also capture traffic data attributes that are not used by the detection algorithm. This results in high volumes of superfluous data which must be removed before analysis. The information in Table 2 will be used in Chapter 4 as a factor in determining which fields should be included in the IPFIX template.

The next chapter examines how IPFIX template extensibility, variable length fields and support for EEs can take botnet traffic capture from a big data challenge to a manageable data solution, by creating BotProbe, a template for botnet traffic capture.

4
BotProbe: A Novel IPFIX Template for Botnet Traffic Capture

4.1 Introduction

Chapter 3 reviewed many of the key design characteristics of IPFIX over its proprietary predecessor NetFlow. *Gates, et al.* (2004) considered NetFlow v5 to be inefficient at traffic capture for security analysis, with only 10 of the 18 fields in the fixed template configuration of NetFlow v5 capturing attributes pertaining to network security. *Velan* (2013) concurred, stating that the lack of template customisation in NetFlow v5 and NetFlow v9, make them limited for use in threat detection.

Until now, botnet detection algorithm design has been confined to the few traffic attributes that can be acquired through NetFlow. When a researcher needs to capture an attribute that is not contained within the 18 NetFlow v5 fields, PCAP can supplement data capture (*Sperotto, et al., 2010*). In high-speed data networks, such as a CSP environment, such inefficiencies from PCAP quickly translate into big data traffic volumes that burdens both analysis and storage. Some researchers have attempted to address the excessive traffic volumes captured by PCAP by creating packet capture aggregation procedures (*Narang, et al., 2014; Hang, Wei and Faloutsos 2013; Narang, Reddy and Hota, 2013*). Aggregation reduces the traffic captured by duplicate flows, but still captures considerable traffic within the packet body.

This chapter describes how three design features of IPFIX, namely template extensibility, enterprise elements and variable length fields, can address the limitations of NetFlow and PCAP data capture. This opens the path for the creation of new botnet detection algorithms, where researchers now have the ability to dictate the attributes that require capture for their algorithms, rather than the capture mechanism prescribing which attributes can and cannot be used in the detection algorithms.

4.2 IPFIX Template Customisation

The research hypothesis states that IPFIX should offer advantages over NetFlow v5 for the detection of botnet communications in CSPs environments. In order to justify demonstrable advantages, an IPFIX template should be able to a) capture botnet traffic characteristics that cannot be captured by NetFlow, b) capture botnet traffic more efficiently than NetFlow or c) identify botnet traffic earlier in the botnet life cycle than NetFlow allows.

Of the seven features that the IPFIX protocol can offer for botnet traffic capture over NetFlow, as outlined above in Chapter 3, *Santos* (2016) claims that the most important design enhancement of IPFIX is template customisation. Customisation means an IPFIX template is no longer confined to capturing the 18 fields that NetFlow v5 captures, of which 8 fields are superfluous to security threat detection (*Gates et al., 2004).* In a customised IPFIX template each field element can be fully utilised in the detection of a botnet communication traffic. This allows the creation of an IPFIX flow Protocol Data Unit (PDU) that can capture the same, or more, information than a NetFlow flow PDU, a less than the 48 byte size of a NetFlow v5 PDU. In a high-speed data networks such as a CSP, reducing the size of the flow PDU should in turn reduce both device processing power requirements and data storage requirements.

Template customisation can be described as three features:

(1) Template extensibility allows the creation of customisable export templates, in which any of the 433 IEs defined by IANA can be added or remove from the template as required;

(2) Enterprise elements allow the creation of new template fields to export any layer 2 to layer 7 information held within a PDU, should this information not be supported in the 433 IANA defined fields. It is the support for EEs that makes IPFIX superior to NetFlow in next-generation networks (*Velan, Jirsik and Čeleda, 2013)*;

(3) Variable length fields permit efficient capture of data that are not fixed length strings, such as HTTP GET or SMTP Hello messages.

Each of these features contributes to advantages over NetFlow for botnet communication detection. Template extensibility allows the construction a template without superfluous fields which should be smaller than a 48 byte NetFlow v5 PDU. Template extensibility with support for EEs, permit the capture of additional

information over and above NetFlow. Variable field length elements allows capture of application layer traffic that may be able to detect botnets earlier in their life cycle. Many researchers, as outlined in Chapter 3, argue that application layer information is required for the early detection of botnets. The limited subset of attributes in NetFlow v5, predominantly IP address and protocol information, limits botnet detection to the attack phase when a bot is producing the highest number of detectable packets.

It should be noted that NetFlow v9 also supports template extensibility. Two studies into NetFlow v9 for botnet traffic capture (*Wijesinghe, Tupakula and Varadharajan, 2015; Haddadi, et al., 2014*) have been undertaken. However, these studies have failed to recognise the full potential of next-generation flow protocols because the implementation of template extensibility in NetFlow v9 is more rigid than in IPFIX. This rigidity imposes a limitation upon the usefulness of NetFlow v9 in threat detection, as there is less capability for packet content analysis (*Velan, Jirsik and Čeleda, 2013).*

This chapter, and subsequent chapters, makes comprehensive reference to IPFIX IE and EE fields. IANA is responsible for maintaining the list of IPFIX IE name descriptors. IANA does not maintain the list of EEs, as this is the responsibility of the element creator. To ensure a consistent single naming convention throughout this thesis, the nomenclature used for both IE and EE element names is the *SuperMediator* element name descriptors. SuperMediator is an IPFIX collector chosen for this research project, where justification for selection is detailed in Chapter 5, below.

4.3 BotProbe IPFIX Template Creation Methodology

The following section outlines the methodology for creating IPFIX templates for botnet traffic capture. This includes justification for the selection of the botnet data samples used in this study, as well as detailing the equipment and methods used in data capture and analysis.

4.3.1 DATASET

The dataset used throughout this study comes from a malware repositiory maintained by Czech Technical University (CTU), Prague[4]. The CTU repository holds almost 200 botnet samples, collected from 2013 onwards. The datasets have been captured and maintained through academic funding received by CTU.

[4] https://stratosphereips.org/category/dataset.html

Other botnet repositories do exist. Shiravi et al. (2012) presented a study of available botnet datasets and described the properties a dataset should have in order to be used for comparison purposes. *García, Uhlíř and Rehak* (2014) considered Shiravi's work to explain the weaknesses of several publically available datasets such as the CAIDA, DARPA and KDD datasets to justify using datasets from the CTU repository. They found that anonymisation of payloads in public datasets has an impact on research output. Several commercial organisations maintain their own malware datasets. However, these tend to be proprietary and are not readily available to academic researchers. Public repositories suffer from two drawbacks, a) they are often maintained by individuals so tend not to be as up to date or extenive as the CTU datasets and b) samples have often not been anoymised to respect confidentiality of personal data, which is against the ethical considerations of this study (Chapter 1.5).

Alternatively, it would have been possible to create our own bespoke datasets for this study. The fundamental reason for choosing not to do this was the limited ethical availability of legitimate bot C&C server software. C&C servers are required to construct the bot executable software needed to create reliable, replicable botnet samples. Only four C&C server softwares could be found through legitimate sources; Zeus, Spybot, Spyeye and Mirai; thus reducing variablilty in generated test data. Instead, these softwares were used in validation testing of the concept build in Chapter 6, as the C&C software allowed complete control of the malware during testing.

The CTU repository was chosen as the dataset for this study for several reasons. Reliability of the datasets in this repository can be considered high, as CTU provide ground truth justification for each individual dataset, including VirusTotal analysis outputs for each sample. Variance across dataset samples is ensured as the repository is one of the largest and most varied collections of botnet samples held in PCAP format. Thus maintaining generalisability of sample data from a single repository, rather than requiring datasets from multiple repositories. The availability of these datasets to the research community ensures replicability to researchers to confirm the results in this study using the same datasets. Data accuracy can be assumed as the CTU dataset has been used in other academic studies. *García and Pechuocek* (2016) used CTU122_1(Geodo) and CTU162_1(Upatre) to study connectity between C&C servers. *Kirubavathi and Anitha* (2016) used Kazy, Medfos, Kelihos and Sogou datasets to study flow characteristics, although did not quote the specific dataset reference numbers. *Haddadi and Zincir-Heywood* (2015) used Virut, NSIS, ZeroAccess and Kelihos samples to study connection pattersn between HTTP bots. Again they did not specificy dataset sample references numbers. *Sangroudi and*

Mirabedini (2015) used 13 different CTU datasets to study the use of fuzzy clustering in botnet detection. *García, Uhlíř and Rehak* (2014) used CTU44 through to CTU54 to study C&C bots characteristics in NetFlow. Whilst these authors did not use the exact same dataset samples as this study, CTU is considered to be a trustworthy and reliable repository.

The CTU repository holds almost 200 botnet samples. To simulate a random selection process, samples were arbitarily selected from the CTU repository with no perceived bias. The criteria for a sample was that it must comprise over 100,000 flows over a range of communication channel protocols. In test samples of less than 100,000 flows, it was found that bot information was lost in background noise. As this criteria reduced the number of bot samples available for testing, no restriction was placed upon the creation date of the sample. Selected test samples are listed in Appendix B.

4.3.2 EQUIPMENT

The test samples in the CTU repository are presented in PCAP format. PCAP will be fed into an IPFIX exporter and then the exporter will extract botnet traffic. The predominant IPFIX exporters that support PCAP as an input stream are nProbe and YAF (Yet Another Flowmeter). Either nProbe or YAF could have been selected as the exporer in this test, as both support a wide range of IEs and EEs for data traffic capture; as summarised in Table 3. nProbe supports 69 IEs compared to 43 IEs by YAF, although many of the addition IEs supported by nProbe collect flow contextual statistics. YAF supports a larger range of traffic contextual EEs (refer to Table 4). Chapter 5 compares six IPFIX exporters, and provides further justification for the selection of YAF as the chosen IPFIX exporter for this research. YAF exports IPFIX as a proprietary `.yaf` format. An IPFIX mediator is required to convert this into a `.csv` format suitable for statistical analysis packages. This study used SuperMediator. Again, refer to Chapter 5 for justification for the selection of SuperMediator as the IPFIX mediator software for flow collection.

TABLE 3. SUPPORT FOR IANA DEFINED IES AND EE PROTOCOLS, BY IPFIX PROBE

Probe	IEs	EE Protocols
nProbe (*Deri, 2003*)	69	*BGP, DHCP, DNS, FTP, GTP, HTTP, IMAP, MySQL, Oracle, POP, Radius, RTP, STP, SMTP*
YAF / SuperMediator (*Inacio and Trammell, 2010*)	43	*DHCP, DNP, DNS, FTP, HTTP, IMAP, IRC, MySQL, NNTP, POP, RTP, SIP, SMTP, SSH, SSL, TFTP*

TABLE 4. SUPPORT FOR EES IN BOTNET TRAFFIC PROTOCOLS, BY IPFIX PROBE

Probe	DNS	FTP	HTTP	IRC	SMTP	TFTP
nProbe	6	4	9	-	2	-
YAF / SuperMediator	16	5	45	1	11	2

The test environment for the template construction comprised a single host server running a single guest VM. The host server was a Dell Latitude E5440 laptop, with an Intel i5-4310U CPU 2.6GHz and 8GB RAM, running Windows 7 Enterprise 64-bit SP1 and VMWare Workstation 11.1.2. The guest VM was a Ubuntu 14.04 LTS desktop configured with four 2.6GHz processors and 2.9GB RAM, into which YAF v2.8.4 and SuperMediator v1.3.0 were installed.

4.3.3 METHOD

The aim of this test was to quantify the capability of each template element to capture botnet characteristics within flow traffic. This would facilitate construction of an IPFIX template where each template field has been proven to be statistically significant in the detection of botnet communication traffic. The independent variables in this test were the individual IEs and EEs available to the exporter template. The dependent variables were the botnet samples from the CTU repository. Figure 6 summarises the test method.

The configuration for exporting IEs ("`ie_tester.conf`") differs slightly to the configuration for exporting EEs ("`ee_tester.conf`"). With IE export, SuperMediator requires IE IDs to be specified after the **`FIELDS`** keyword under the exporter declaration. All flows are output to a single `.csv` file. With EE export, a second exporter is declared using a **`DPI`** path. The EEs to be exported have their IDs defined in tables within the **`DPI_CONFIG`** block. Each defined table outputs its own `.txt` file, titled as per the table name.

The detailed IPFIX template creation method was:

(1) A bot sample was selected, at random, from the CTU repository;

(2) YAF was configured to convert the `.pcap` sample into `.yaf` IPFIX format:

```
# yaf --in in_file.pcap --out out_file.yaf -v
--plugin-name=/usr/local/lib/yaf/dpacketplugin.la
--applabel --max-payload 65535
```

(3) The IPFIX stream ("`out_file.yaf`") was fed into SuperMediator, which exports the fields defined in "`ie_tester.conf`" to a `.csv` file:

```
# super_mediator --config ie_tester.conf
```

(4) The `.csv` file output from SuperMediator was analysed as per Figure 7 below.

SuperMediator configuration files can be found in Appendix C.

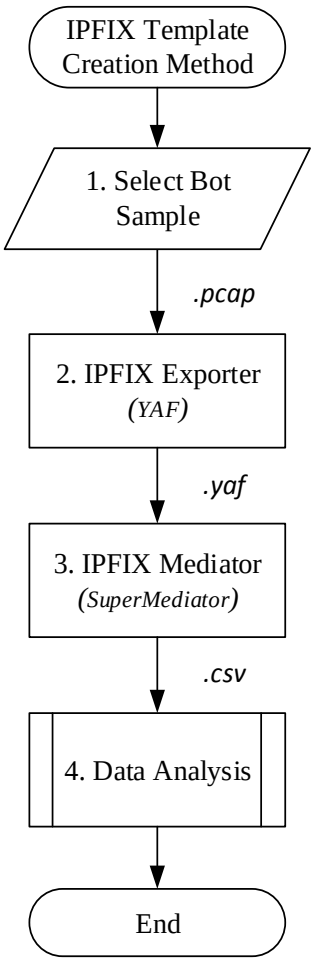


Figure 6. Flow diagram of the IPFIX template creation test.

4.3.4 ANALYSIS

IPFIX IEs capture both situational data about a flow, such as *maxFlowEndMicroseconds* (IANA_ID#268) or *flowSamplingTimeSpacing* (IANA_ID#399), and traffic contextual data about a packet, such as *protocol* (IANA_ID#4) or *source port* (IANA_ID#7). The EEs defined in SuperMediator, tend to be more traffic contextual based.

Botnet detection through communication traffic relies more upon information contained within traffic contextual data, rather than within flow situation data. YAF/SuperMediator presented 75 IEs, of which 43 are recognised by IANA, and 288 EEs as test candidates for botnet traffic indicators. Each IE and EE was tested for two factors for inclusion in the template:

- Quantity - the overall occupancy of the test element in botnet traffic;
- Quality - the content of that element field being present in botnet traffic, but being different to other elements being captured.

Quantity was measured through frequency analysis. A low field count meant that the element was not present in sufficient quantities in botnet traffic to justify collection. Visual inspection of the frequency data defined the low quantity cut-off threshold as when < 1% of fields were either consistently empty, or contained null values which had no meaning in the data.

Quality was measured through correlation analysis. Correlation to other elements determines which elements are retained within the template. Cohen's classification was used to interpret effect size: small (<0.10), medium (<0.30) and large (<0.50) (*Cohen, 1988*), as this allowed analysis to focus on the smaller effect correlations. Where the correlation with other elements was interpreted as large, this indicated that the two test elements captured data that was similar, or identical; therefore one of the duplicating elements could be discarded in order to maintain template efficiency. Where the correlation with other elements was interpreted as smaller, this indicated that this test element captured data that was not being captured by other elements.

Low duplication does not, in itself, guarantee inclusion within the template. To be included, the element must display a high quantity (as defined above) and either academic literature could be found confirming the usefulness of the element as a botnet traffic classifier, or visual inspection and intuition of the captured data indicates the element may have usefulness as a botnet classifier.

Scatter plots were also used to indicate the strength of a linear, or curvilinear, relationship between two continuous variables (*Pallant, 2013*). These were plotted prior to correlation, as a general indication of variable relationship with which to compare. Hierarchical clustering of elements could also have indicated similarity between IEs and EEs, but was inconclusive across this dataset.

In large sample sizes, visual inspection of frequency distribution histograms are usually a strong enough indicator of linearity of the sample distribution (*Collins, 2014; Field, 2009*; *Altman and Bland, 1995*). The sample sizes in this study were considered large at n > 100,000. However, this approach is not always reliable (*Field, 2009*). Statistical techniques are available for measuring how a distribution appears to differ from a normal distribution; including the Shapiro-Wilk test, the D'Agostino skewness test, and the Anscombe-Glynn kurtosis test (*Ghasemi and Zahediasl, 2012*). However, in large populations, Kolmogorov-Smirnov (K-S) with 95% confidence levels are used for distribution testing (*Dancey and Reidy, 2007).* K-S is sensitive to outliers, so Lilliefors correction can be used to make this test less conservative (*Peat and Barton, 2005*). In K-S testing, H_0 assumes the sample demonstrates normal distribution and H_1 assumes the sample is significantly different to normal distribution. In a one-sample K-S test, with α set to 0.05, when $\rho < \alpha$, H_0 is rejected.

Correlation coefficients are used to estimate the degree of association between two variables. Pearson's product-moment correlation coefficient (PPMCC) estimates the strength of a linear relationship when the test variables are either ratio or interval, and approximate a normal distribution. Although PPMCC is not a meaningful figure if it has been obtained from a sample which shows any curvilinear relationship (*Clegg, 1995*). Instead, Spearman Rank Order Correlation (Spearman's rho) is used when one or both variables are ordinal (*Pallant, 2013*), or when sample data is not normally distributed (*Greenhalgh, 1997*). Spearman's rho is able to measure both linear and non-linear relationships, as it is unaffected by sample distribution (*Gauthier, 2001*). Spearman's rho is also less prone to outliers than PPMCC (*Gauthier, 2001*). With large enough sample sizes, any violation of normality assumption does not cause major problems (*Pallant, 2013*). In Spearman's rho testing, H_0 assumes there is no correlation between variables A and B, H_1 assumes either a direct (positive) correlation or an indirect (negative) correlation.

Spearman's rho was used on all data in this study, as this study a) combined discrete with ordinal data, and b) contained data that closely approximates a normal distribution as well as data that strongly demonstrates a non-normal distribution. Alternatively, PPMCC could have been used for normally distributed data and

Spearman's rho for data that did not demonstrate normal distribution. However, the use of two varying analysis methods would have made direct comparison of the correlation coefficients less effective. Flow diagram Figure 7 summarises the data analysis method.

4.3.5 ANALYSIS METHOD

The detailed data analysis method was:

(1) Each traffic contextual IE and EE were selected, in turn, for analysis;

(2) The field count frequency was recorded for each element in the botnet sample;

(3) Low occupancy fields, defined as <1.0% of fields containing data, were discarded under the basis of template space optimisation, in that they do not capture sufficient data to justify the space they require in the template;

(4) Data was categorised and cleansed before correlation takes place:

- The test element was categorised as either nominal (discrete named data, i.e. string or URL), interval (continuous numeric variable) or ratio (continuous numeric variable where "0" has a meaning);
- "0" values in interval data are considered as null and therefore removed;
- Nominal data was transformed to ranked numerical data.

(5) Normality distribution indicated the use of parametric or nonparametric statistical analysis methods. With large sample sizes (n > 10,000) the normality assessment was statistically confirmed using Lilliefors corrected K-S tests;

(6) Scatter plots confirm the normality assessment in the above step;

(7) Correlation testing was used to calculate the effect size between variables. When all data showed normal distribution, Pearson Product-Moment Correlation Coefficient was used. Where normality distribution showed both normally and non-normally distributed data in a population, Spearman's rho correlation was used to allow direct comparison of coefficients. Two-tailed analysis was used to identify the correlation direction;

(8) Cohen's classification was used to interpret the effect size and the element was either retained in the template (9) or discarded (10).

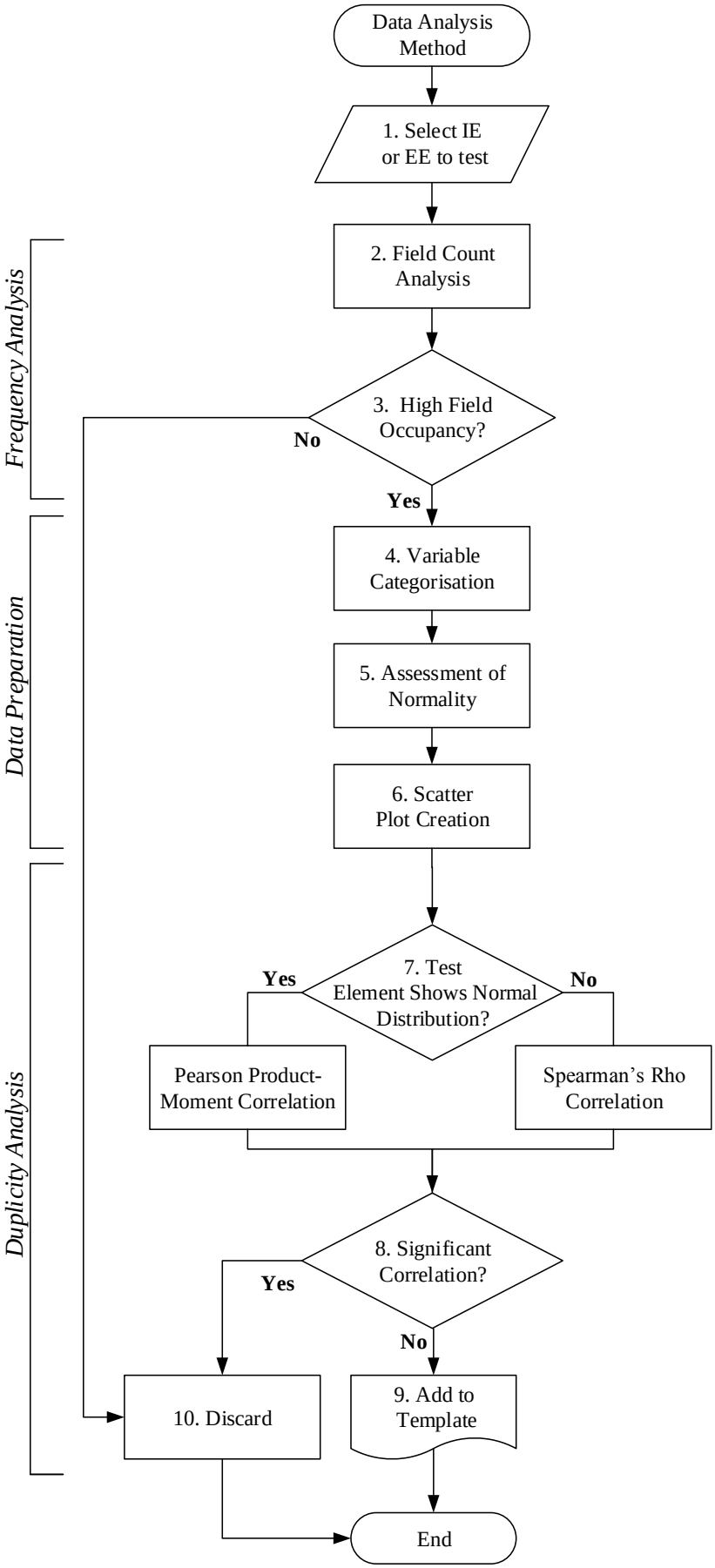


Figure 7. Flow diagram of the IPFIX template creation data analysis.

4.4 Information Elements Results

4.4.1 FREQUENCY ANALYSIS

YAF/SuperMediator supported the capture of a total of 75 IPFIX IEs. Field occupancy analysis of the 21 bot samples (refer to Table 7) revealed that 44 IEs consistently captured no data at all. So these 44 IEs were removed from further analysis.

Of the 31 remaining IEs, a further six IEs were also removed from further analysis:

- ***sTime*** and ***eTime*** - these IEs contained identical data to flow start and end fields ***sTimeMS*** and ***eTimeMS*** respectively;
- ***sIP_INT*** and ***dIP_INT*** - these IE are simply the integer equivalent value of IPv4 address fields ***sIP*** and ***dIP*** respectively;
- ***DPI*** and ***flowKeyHash*** - are used when capturing EEs, so are ignored until the EEs are analysed, below.

The total field count of the remaining 25 IEs is displayed in Table 5. A breakdown of field count by individual bot sample can be found in Table 31 in Appendix A.

TABLE 5. FIELD COUNT AND DATA TYPE CATEGORISATION FOR THE 25 IES ANALYSED DURING CONSTRUCTION OF THE BOTPROBE TEMPLATE (BOT SAMPLES = 21; FLOW RECORDS = 7,363,521)

INFORMATION ELEMENTS	sIP	dIP	sPort	dPort	protocol	application	duration
SuperMediator_ID#	0	1	4	5	6	7	17
Total Field Count	7363521	7363521	7363521	7363521	7363521	4383033	7363521
Occupancy	100.0%	100.0%	100.0%	100.0%	100.0%	59.5%	100.0%
Nominal/Interval/Ratio	N	N	R	R	R	I	R

sTimeMS	eTimeMS	packets	rPackets	bytes	rBytes	iFlags	rIFlags	uFlags
20	21	25	26	27	28	29	30	31
7363521	7363521	7363521	5620371	7363521	5620371	7363521	7363521	7363521
00.0%	100.0%	100.0%	76.3%	100.0%	76.3%	100.0%	100.0%	100.0%
R	R	R	I	R	I	R	R	R

rUFlags	attributes	rAttributes	tcpSeq	rTcpSeq	endReason	ToS	rToS	collector
32	33	34	37	38	41	75	76	80
7363521	891114	347368	7357619	7363521	7363521	19583	4	7363521
100.0%	12.1%	4.7%	99.9%	100.0%	100.0%	0.3%	0.0%	100.0%
R	I	I	R	R	R	I	I	N

4.4.2 VARIABLE CATEGORISATION

The variable categorisation of each of the 25 IEs prior to transformation are provided in Table 5. All nominal elements were transform to numerical data for correlation:

- ***sIP,*** nominal [IP address = X.X.X.X] to ratio[5];
- ***dIP:*** nominal [IP address = X.X.X.X] to ratio;
- ***collector:*** nominal [Collector name = "c1"] to ratio.

Data was cleansed to maximise potential correlations. "0" values in interval variables were deleted, because "0" indicates blank data in the following fields:

- ***application***;
- ***rPackets****;*
- ***rBytes***;
- ***ToS****;*
- ***rToS***;
- ***attributes****;*
- ***rAttributes***.

4.4.3 ASSESSMENT OF NORMALITY

Across all the IEs tested, Lilliefors corrected K-S tests were typically <0.001; indicating that the samples did not demonstrate normal distribution. Q-Q plots and histograms of the elements also displayed non-normal distribution, thereby confirming the use of Spearman's rho for correlation analysis. This is common in large samples (*Pallant, 2013*) where assessing normality becomes less important.

These findings agree with Collins who states that finding "parametric distributions in raw network data are rarer that the Yeti" (*Collins, 2014*).

[5] SuperMediator also exports *sIP_INT* and *dIP_INT* IEs, which are integer values of IPv4 addresses *sIP* and *dIP*, but do not convert MAC addresses. It was found to be more reliable to ignore *sIP_INT* and *dIP_INT* and use transformed values of *sIP* and *dIP.*

4.4.4 SCATTER PLOTS

Figure 8 shows scatter plot matrices for two bot samples selected at random, CTU3_1(Kelihos) and CTU25_1(Zbot), where r_s is Spearman's rho correlation, ρ indicates the probability of the correlation through chance and n is the total number of flows correlated in that sample. The scatter plots generally varied in shape between IEs, whilst retaining a similar structure when viewed for the same IE across different bot samples. Chapter 4.7.1 contains a discussion on Figure 8.

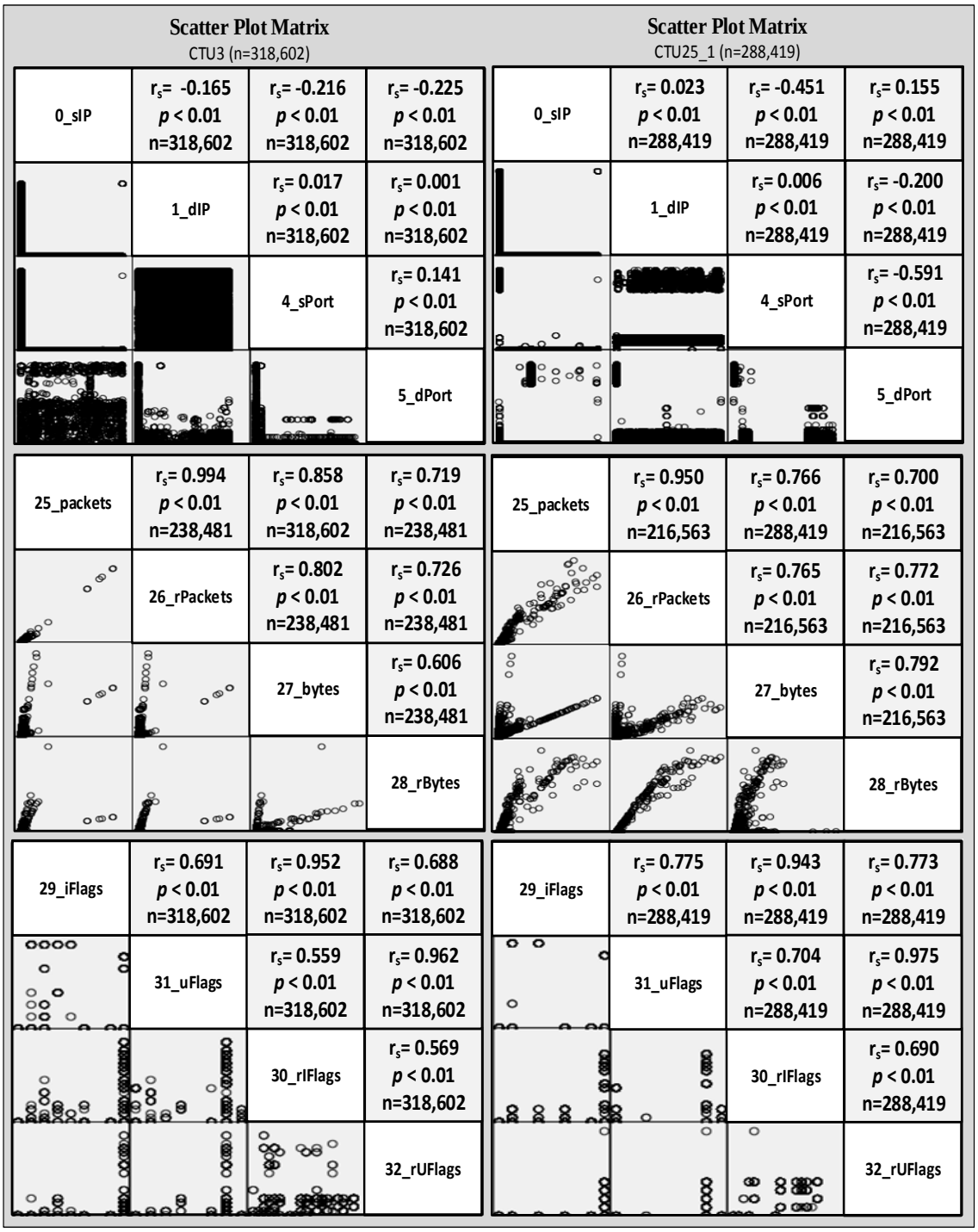


Figure 8. IE scatter plots.

4.4.5 CORRELATION TESTING

The assessment of normality analysis, above, indicated that all IEs demonstrated a non-normal distribution and would therefore undergo comparison via Spearman Rank Order Correlation. Each IE within a test sample was correlated with all the other IEs in that sample. To allow comparison of relative correlation effect sizes across all 21 test samples, each correlation matrices for all 21 samples were aggregated into the single correlation matrix in Table 6. Table 7 summarises the 21 test samples used to create Table 6. In order to allow visual interpretation of the correlation effects across the sample population of 21 test samples, each IE in Table 6 was colour coded according to the overall effect size score. For example, Table 6 shows that ***sPort*** correlated with ***dIP*** with an r_s >=0.5 for 2 samples, r_s >=0.3 for 11 samples, r_s >=0.1 for 6 samples, and hence 2 samples had r_s <0.1, giving an overall effect size score of 39, which equates to a medium correlation hence the matrix square is coloured yellow. The score for each effect size was:

- Each correlation >= 0.500 was assigned 5 points;
- Each correlation >= 0.300 was assigned 3 points;
- Each correlation >= 0.100 was assigned 1 point;
- Each correlation < 0.100 was assigned -5 points.

These effect score points are arbitrary, and simply provide a relative scale for comparison. Because smaller effect sizes are of interest to this study, the points scale was engineered to match correlation values, where larger correlations were assigned a higher score to ensure the lower scores are visible. This provided suitable granularity to highlight any anomalous samples that differ radically from other samples arising due to differences in the behaviour of each bot tested. An alternative method is to correlate all 7 million flows into a test. This was rejected because small correlation differences were dwarfed by the much larger effects within the overall population.

Table 6 demonstrates several areas of correlation clustering. The most obvious cluster is amongst the TCP flags; when orange squares indicate a large correlation. TCP flags also exhibit medium strength clusters with port/protocol/application, and with packets/bytes. A medium cluster is shown between packets/bytes/protocols/ports; and another medium cluster between IP/ports. This becomes important when constructing the final IPFIX template as it suggests that some IEs need not be exported because their behaviour is captured through other IEs. Observed IE relationships are discussed and justified in Chapter 4.7.1.

TABLE 6. AGGREGATED CORRELATION MATRIX: IES
(BOT SAMPLES =21; FLOW RECORDS =7,363,251)

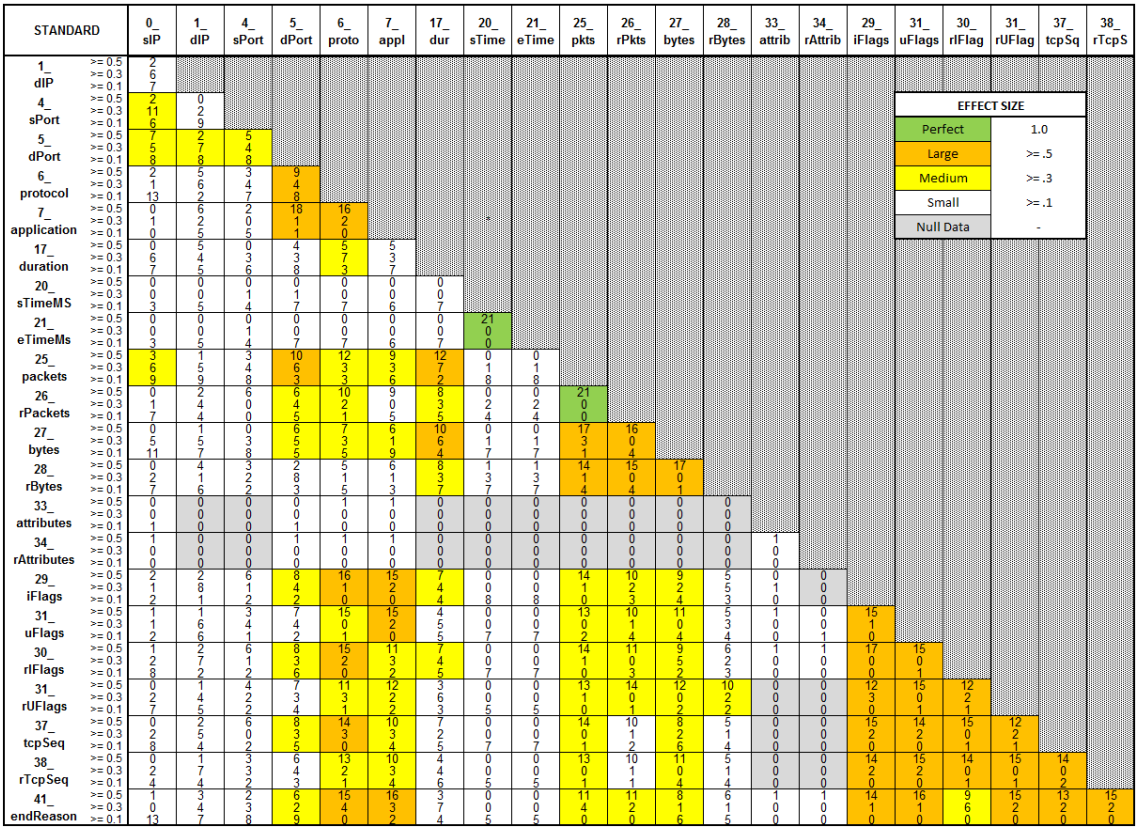


TABLE 7. 21 BOTNET SAMPLES USED IN THE CREATION OF THE IE AGGREGATED CORRELATION MATRIX

Sample	Sample Date	Bot (VirusTotal)	Flows
CTU3_1	21/07/2013	Kelihos	318,602
CTU8_1-win5	10/09/2013	Zbot	186,958
CTU8_1-win9	10/09/2013	Zbot	168,065
CTU10_1-win7	11/07/2013	Unknown	178,564
CTU10_1-win9	11/07/2013	Unknown	278,687
CTU10_1-win10	11/07/2013	Unknown	231,420
CTU16_1-win5	23/08/2013	Kelihos (Waledac)	812,996
CTU16_1-win11	23/08/2013	Kelihos (Waledac)	801,931
CTU25_1	09/09/2013	Zbot	288,419
CTU25_5	10/02/2014	Zbot	829,624
CTU110_4	09/04/2015	HTbot	284,196
CTU144_1	23/09/2015	Shifu	408,482
CTU145_1	23/09/2015	Fake uTorrent	411,928
CTU148_1	26/09/2015	Zusy	172,287
CTU149_1	05/12/2015	Kelihos	235,287
CTU149_2	09/12/2015	Kelihos	207,959
CTU160_1	29/04/2016	Tinba (Andromeda)	206,008
CTU165_1	27/05/2016	Zeus (New Variant)	230,475
CTU166_1	29/04/2016	Tinba (Andromeda)	583,368
CTU167_1	27/05/2016	Storm	197,941
CTU168_2	03/08/2016	Andromeda	330,696

4.5 Enterprise Elements Results

4.5.1 FREQUENCY ANALYSIS

YAF/SuperMediator supported the capture of a total of 288 individual EEs[6]. Field occupancy analysis of the 33 bot samples (Tables 15, 17, 19 and 21) revealed that 225 EE consistently captured no data, so these 225 were removed from further analysis.

The remaining 63 fields that exported data, fell across nine protocols: NNTP, RTP, SIP, SSH, HTTP, DNS, SMTP, SSL and IRC. The distribution of all nine protocols, across all botnet samples (n=17,192,796 flows) are provided in Table 8.

Of these nine protocols, HTTP, DNS, SMTP and SSL captured 99.8% of the overall traffic and comprised the focus of this study (Tables 9 - 12). Although IRC exhibited a low traffic count, as IRC is still a commonly used bot communication channel, IRC was retained within the study data. One explanation for the lower than expected overall traffic percentage of IRC could be because the sample bots were relatively recent, dating from 2013 to 2016. For a breakdown of field count by individual bot sample, see Appendix A: Table 32 (HTTP), Table 33 (DNS), Table 34 (SMTP), Table 35 (IRC) and Table 36 (SSL).

The remaining 0.2% of traffic comprised NNTP, RTP, SIP and SSH. No evidence could be found in literature to suggest that NNTP, RTP, SIP, or SSH have been used as bot communication channels. Rather than dedicate template space to these four protocols, ***sPort*** and ***dPort*** IEs can be used to indicate if a flow contains any of these protocols through their corresponding port number.

TABLE 8. PROTOCOL DISTRIBUTION FOR THE 33 BOTNETS SAMPLED

Protocol	Flows	% of overall traffic
NNTP	4	0.0%
RTP	38,407	0.2%
SIP	32	0.0%
SSH	102	0.0%
HTTP	7,167,557	41.7%
DNS	8,655,304	50.3%
SMTP	877,827	5.1%
SSL	453,303	2.6%
IRC	260	0.0%

[6] https://tools.netsa.cert.org/super_mediator/super_mediator.conf.html (see "DPI configuration Block")

TABLE 9. FIELD COUNT AND DATA TYPE CATEGORISATION FOR HTTP EES
(BOT SAMPLES = 17; HTTP FLOW RECORDS = 7,167,557)

HTTP	Server String	User Agent	Get	Version	Referer
SuperMediator_ID#	110	111	112	114	115
Total Field Count	51044	930256	955053	977373	56619
Occupancy	0.7%	13.0%	13.3%	13.6%	0.8%
Nominal/Interval/Ratio	N	N	N	N	N

Location	Host	Content Length	Age	Accept	Accept Lang	Content Type
116	117	118	119	120	121	122
225810	672754	712180	9957	671085	141112	710588
3.2%	9.4%	9.9%	0.1%	9.4%	2.0%	9.9%
N	N	R	R	N	N	N

Resp.	Cookie	Set Cookie	Auth.	Via
123	220	221	252	253
946977	31592	65905	5095	4157
13.2%	0.4%	0.9%	0.1%	0.1%
N	N	N	N	N

TABLE 10. FIELD COUNT AND DATA TYPE CATEGORISATION FOR DNS EES
(BOT SAMPLES = 15; DNS FLOW RECORDS = 8,655,304)

DNS	A Record	NS Record	CNAME Record	SOA Record	MX Record	PTR Record	TXT Record	AAAA Record
SM_ID#	1	2	5	6	12	15	16	28
Total	4653441	722218	60868	157665	1290	699362	494193	447267
Occupancy	53.8%	8.3%	0.7%	18.2%	0.0%	8.1%	5.7%	5.2%
N / I / R	N	N	N	N	N	N	N	N

TABLE 11. FIELD COUNT AND DATA TYPE CATEGORISATION FOR SMTP EES
(BOT SAMPLES = 4; SMTP FLOW RECORDS = 877,827)

SMTP	Hello	From	To	Content Type	Subject	Response	Rcvd Date
SuperMediator_ID#	162	163	164	165	166	169	251
Total Field Count	197758	122212	197732	35920	84389	197754	42062
Occupancy	22.5%	13.9%	22.5%	4.1%	9.6%	22.5%	4.8%
Nominal/Interval/Ratio	N	N	N	N	N	N	N

TABLE 12. FIELD COUNT AND DATA TYPE CATEGORISATION FOR SSL EES
(BOT SAMPLES = 12; SSL FLOW RECORDS = 453,303)

SSL	Comm. Name	Private Org.	Country Name	Locality Name	State Name
SuperMediator_ID#	3	5	6	7	8
Total Field Count	59651	1111	63907	17572	16846
Occupancy	13.2%	0.2%	14.1%	3.9%	3.7%
Nominal/Interval/Ratio	N	N	N	N	N

Street Address	Org.	Org. Unit	Private Org.	Postal Code	Client Version	Server Cipher
9	10	11	15	17	186	187
943	64134	30042	879	989	11284	11284
0.2%	14.1%	6.6%	0.2%	0.2%	2.5%	2.5%
N	N	N	N	N	I	N

Cert Version	Cert Serial	NotValid Before	NotValid After	PublicKey Length	Record Version
189	244	247	248	250	288
33002	33002	33002	33002	31370	11284
7.3%	7.3%	7.3%	7.3%	6.9%	2.5%
I	N	N	N	I	N

TABLE 13. FIELD COUNT AND DATA TYPE CATEGORISATION FOR IRC EES
(BOT SAMPLES = 3; IRC FLOW RECORDS = 260)

IRC	TextMsg
SuperMediator_ID#	125
Total Field Count	260
Occupancy	100.0%
Nominal/Interval/Ratio	N

4.5.2 VARIABLE CATEGORISATION

The variable categorisation for each of the 51 EEs prior to transformation is provided in Tables 9 - 13. Most EEs were variable length strings and were therefore nominal variables. All nominal elements were transformed to numerical data for correlation.

Data was cleansed to maximise potential correlations. "0" values in interval variables were deleted, as "0" indicates blank or null data in the following fields:

- ***sslClientVersion;***
- ***sslCertificateVersion;***
- ***sslPublicKeyLength***.

4.5.3 ASSESSMENT OF NORMALITY

Across all the EEs tested, Lilliefors corrected K-S tests were typically <0.001; indicating that the samples did not demonstrate normal distribution. Q-Q plots and histograms of the elements also displayed non-normal distribution, thereby confirming the use of Spearman's rho correlation.

4.5.4 SCATTER PLOTS

Scatter plot matrices for HTTP, DNS, SMTP and SSL are shown in Figures 9 -12. Each protocol scatter plot pair were generated from two bot samples selected at random from the test samples for that protocol (Tables 15, 17, 19 and 21) for visual comparison of EEs across different bots. For example, Figure 9, the plot of ***httpUserAgent*** against ***httpGet*** for CTU25_1(Zbot) shows a single agent generating multiple GET requests, whilst the same plot for CTU145_1(uTorrent) shows several agents generating multiple GET requests. Indicating a behavioural difference between the two bots. Chapter 4.7.2 contains a discussion on Figures 9 - 12.

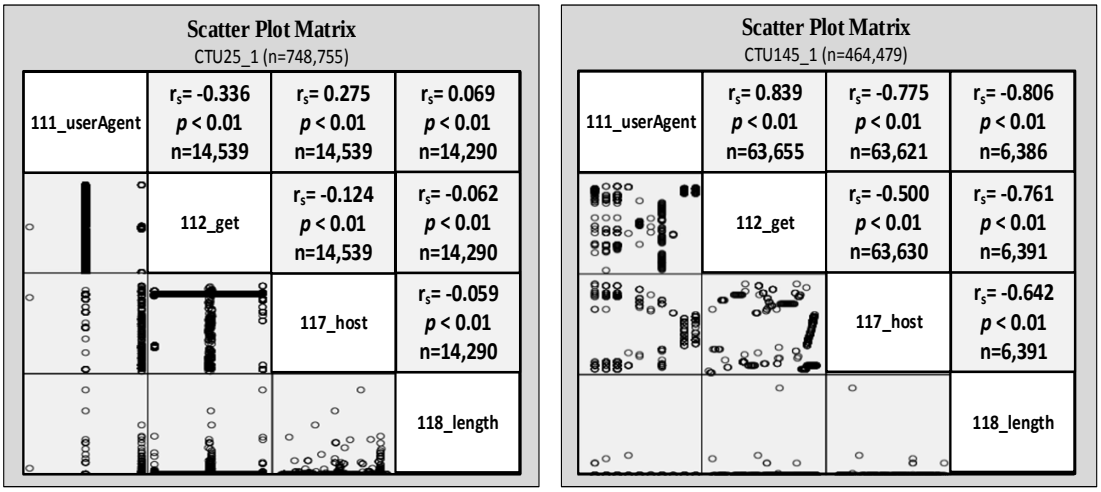


Figure 9. HTTP scatter plots.

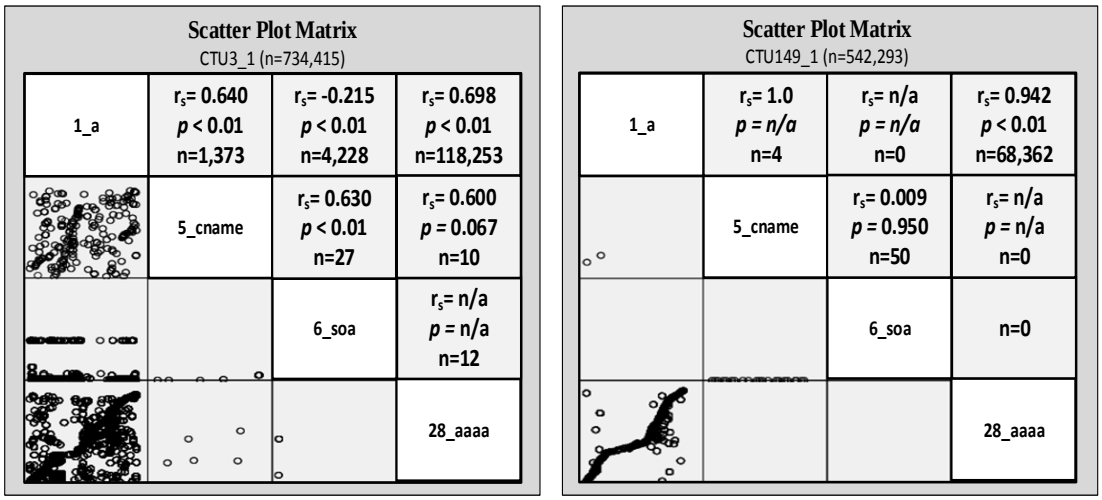


Figure 10. DNS scatter plots.

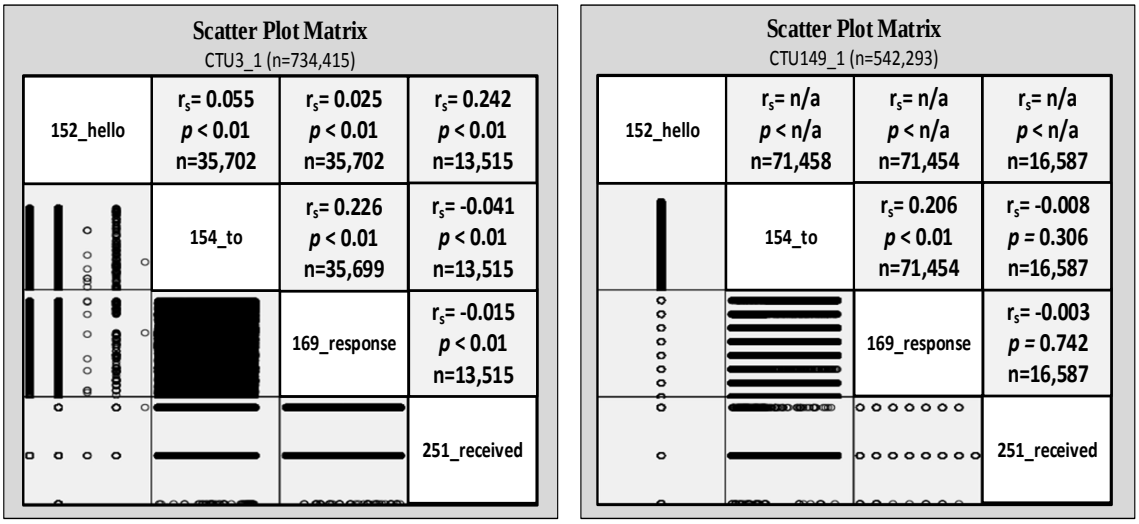


Figure 11. SMTP scatter plots.

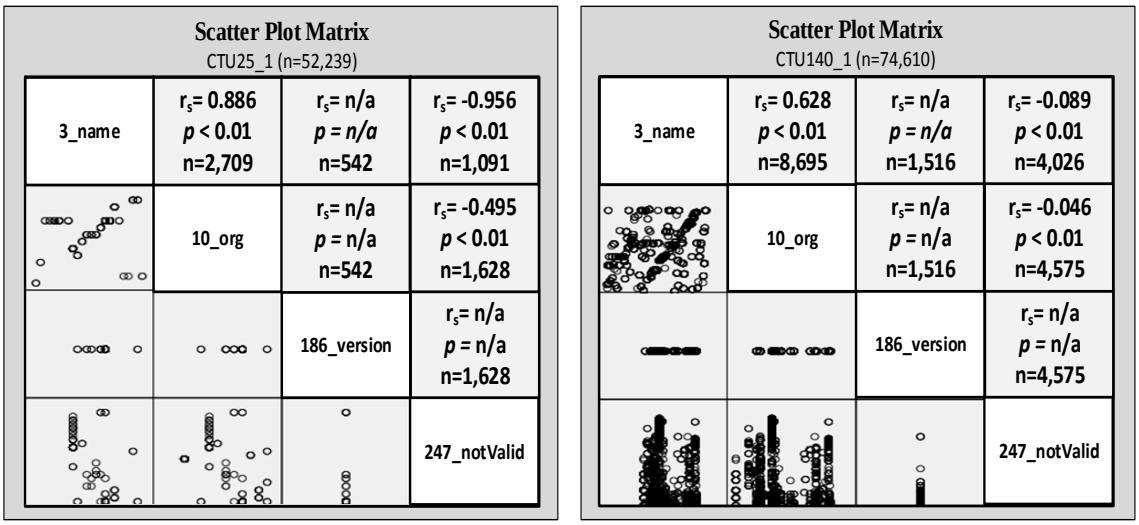


Figure 12. SSL scatter plots.

4.5.5 CORRELATION TESTING

The assessment of normality analysis, above, indicated that all EEs demonstrate non-normal distribution and would therefore undergo comparison via Spearman Rank Order Correlation.

Using the same method as described in Chapter 4.4.5, aggregated correlation matrices for each protocol were created and overall effect size score was colour coded to permit visual comparison of protocol EEs across the test population. For aggregated correlation matrices by protocol, refer to Tables 15, 17, 19, and 21. The bot samples tested to create these matrices are summarised in Tables 14, 16, 18 and 20. Chapter 4.7.2 contains a discussion on Tables 14 - 21.

TABLE 14. AGGREGATED CORRELATION MATRIX: HTTP
(BOT SAMPLES =7; FLOW RECORDS = 2,733,520)

HTTP		110_ Server	111_ UA	112_ GET	114_ Ver	115_ Referer	116_ Loc	117_ Host	118_ Length	119_ Age	120_ Accept	121_ Lang	122_ Type	123_ Resp	220_ Cookie	221_ SCkie	252_ Auth
111_ UA	>= 0.5	2															
	>= 0.3	1															
	>= 0.1	0															
112_ GET	>= 0.5	1	3														
	>= 0.3	2	2														
	>= 0.1	1	2														
114_ Ver	>= 0.5	1	0	0													
	>= 0.3	1	1	0													
	>= 0.1	2	1	3													
115_ Referer	>= 0.5	2	0	0	0												
	>= 0.3	0	0	0	0												
	>= 0.1	1	3	2	2												
116_ Location	>= 0.5	1	1	2	0	2											
	>= 0.3	0	1	0	0	1											
	>= 0.1	3	3	3	3	1											
117_ Host	>= 0.5	2	4	4	0	2	3										
	>= 0.3	3	1	0	1	2	1										
	>= 0.1	1	2	3	2	0	2										
118_ Length	>= 0.5	1	3	3	0	0	3	2									
	>= 0.3	2	2	0	1	3	1	1									
	>= 0.1	3	1	2	2	3	3	2									
119_ Age	>= 0.5	1	0	0	0	0	1	0	0								
	>= 0.3	0	1	0	0	0	1	2	1								
	>= 0.1	0	1	3	0	4	1	3	2								
120_ Accept	>= 0.5	1	0	0	0	0	1	0	0	0							
	>= 0.3	0	0	0	0	0	1	1	0	1							
	>= 0.1	1	4	2	2	3	0	3	3	3							
121_ Lang	>= 0.5	0	2	0	0	0	1	0	0	0	1						
	>= 0.3	0	0	1	0	0	0	0	0	0	2						
	>= 0.1	1	3	2	0	2	0	5	3	2	2						
122_ Type	>= 0.5	1	0	0	0	1	0	0	3	0	0	0					
	>= 0.3	2	1	4	0	1	0	3	3	0	0	0					
	>= 0.1	3	2	2	2	2	2	0	0	5	4	4					
123_ Resp	>= 0.5	0	3	2	0	0	0	2	2	1	0	0	2				
	>= 0.3	2	2	1	0	0	1	1	0	0	1	1	3				
	>= 0.1	3	1	2	1	3	3	1	2	2	1	3	1				
220_ Cookie	>= 0.5	2	0	1	0	0	2	1	2	0	0	0	2	1			
	>= 0.3	1	1	1	0	2	2	0	1	0	1	0	1	0			
	>= 0.1	0	1	3	2	3	0	2	1	1	0	2	3	4			
221_ Scookie	>= 0.5	1	1	1	0	1	2	2	2	1	0	0	3	0	1		
	>= 0.3	0	0	1	0	0	0	0	2	0	0	0	1	2	1		
	>= 0.1	3	3	1	2	3	2	3	1	0	2	2	1	3	1		
252_ Auth	>= 0.5	0	0	0	0	0	1	1	2	0	0	0	0	0	0	0	
	>= 0.3	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	
	>= 0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
253_ Via	>= 0.5	1	0	0	2	0	2	0	0	1	0	0	0	0	2	0	0
	>= 0.3	1	1	1	0	1	1	1	3	1	1	1	1	0	0	1	0
	>= 0.1	0	0	2	1	0	0	2	1	0	0	0	1	2	0	1	0

EFFECT SIZE	
Perfect	1.0
Large	>= .5
Medium	>= .3
Small	>= .1
Null Data	-

TABLE 15. BOTNET SAMPLES USED IN THE CREATION OF THE HTTP CORRELATION MATRIX

Sample	Sample Date	Bot (VirusTotal)	HTTP Flows
CTU3_1	21/07/2013	Kelihos	136,561
CTU25_1	09/09/2013	Zbot	573,606
CTU66_1	07/04/2014	Sality	195,975
CTU110_4	09/04/2015	HTbot	201,895
CTU111_2	09/04/2015	Unknown	193,255
CTU144_1	23/09/2015	Shifu	410,220
CTU145_1	23/09/2015	Fake uTorrent	395,118

The correlation matrices, Tables 14, 16, 18 and 20, were plotted to allow visualisation of EEs relationship strengths. For example, Table 14 shows a cluster of large and medium effect correlations between http EEs ***host, length, server, ua, get*** and ***loc.*** This becomes important when constructing the final IPFIX template as it suggests that some EEs need not be exported because their behaviour is caught through other EEs. Observed EE relationships are discussed and justified in Chapter 4.7.2.

TABLE 16. AGGREGATED CORRELATION MATRIX: DNS
(BOT SAMPLES =5; FLOW RECORDS = 2,507,560)

DNS		1_ A	2_ NS	5_ CNAME	6_ SOA	12_ MX	15_ PTR
2_NS	>= 0.5	4			EFFECT SIZE		
	>= 0.3	0					
	>= 0.1	0			Perfect		1.0
5_ CNAME	>= 0.5	3	0		Large		>= .5
	>= 0.3	0	1		Medium		>= .3
	>= 0.1	0	1		Small		>= .1
6_SOA	>= 0.5	1	0	2	Null Data		-
	>= 0.3	0	0	0			
	>= 0.1	1	0	0			
12_MX	>= 0.5	0	0	0	0		
	>= 0.3	0	0	0	0		
	>= 0.1	0	0	0	0		
15_PTR	>= 0.5	2	4	1	0	0	
	>= 0.3	1	0	0	1	0	
	>= 0.1	0	0	1	0	0	
28_ AAAA	>= 0.5	5	2	1	1	1	2
	>= 0.3	0	1	0	0	0	1
	>= 0.1	0	1	0	1	0	0

TABLE 17. BOTNET SAMPLES USED IN THE CREATION OF THE DNS CORRELATION MATRIX

Sample	Sample Date	Bot (VirusTotal)	DNS Flows
CTU3_1	21/07/2013	Kelihos	1,159,998
CTU66_1	07/04/2014	Sality	169,363
CTU149_1	05/12/2015	Kelihos	801,947
CTU149_2	09/12/2015	Kelihos	743,481
CTU168_2	03/08/2016	Andromeda	719,765

TABLE 18. AGGREGATED CORRELATION MATRIX: SMTP
(BOT SAMPLES =4; FLOW RECORDS =877,827)

SMTP		162_ Hello	163_ From	164_ To	165_ Type	166_ Subject	169_ Resp
163_ From	>= 0.5	1			EFFECT SIZE		
	>= 0.3	0					
	>= 0.1	1			Perfect		1.0
164_ To	>= 0.5	1	0		Large		>= .5
	>= 0.3	0	0		Medium		>= .3
	>= 0.1	0	1		Small		>= .1
165_ Type	>= 0.5	0	0	0	Null Data		-
	>= 0.3	0	0	0			
	>= 0.1	0	0	0			
166_ Subject	>= 0.5	0	0	0	0		
	>= 0.3	0	0	1	0		
	>= 0.1	0	1	0	0		
169_ Response	>= 0.5	1	0	1	0	0	
	>= 0.3	0	0	0	0	0	
	>= 0.1	0	1	3	0	1	
251_ Recvd	>= 0.5	0	0	0	0	0	0
	>= 0.3	0	0	0	0	0	0
	>= 0.1	1	1	0	0	0	0

TABLE 19. BOTNET SAMPLES USED IN THE CREATION OF THE SMTP CORRELATION MATRIX

Sample	Sample Date	Bot (VirusTotal)	SMTP Flows
CTU3_1	21/07/2013	Kelihos	198,319
CTU110_4	09/04/2015	HTbot	96,299
CTU149_1	05/12/2015	Kelihos	304,851
CTU149_2	09/12/2015	Kelihos	278,358

TABLE 20. AGGREGATED CORRELATION MATRIX: SSL
(BOT SAMPLES =7; FLOW RECORDS = 453,303)

SSL		3_ Name	5_ Org	6_ Country	7_ Locality	8_ State	9_ Street	10_ Org	11_ Org	15_ Org	17_ Postcd	186_ Ver	187_ Cipher	189_ CertV	244_ Cert	247_ NotVld	248_ After	250_ KeyLen
5_ Org	>= 0.5	2																
	>= 0.3	2																
	>= 0.1	2																
6_ Country	>= 0.5	0	2															
	>= 0.3	2	0															
	>= 0.1	3	2															
7_ Locality	>= 0.5	0	5	1														
	>= 0.3	0	0	1														
	>= 0.1	4	2	2														
8_ State	>= 0.5	0	2	1	2													
	>= 0.3	1	1	1	1													
	>= 0.1	5	2	3	3													
9_ Street	>= 0.5	3	3	3	4	4												
	>= 0.3	2	0	0	0	0												
	>= 0.1	1	3	2	1	1												
10_ Org	>= 0.5	6	2	0	0	2	4											
	>= 0.3	1	2	0	2	1	2											
	>= 0.1	0	2	4	5	3	0											
11_ Org	>= 0.5	0	2	0	4	1	4	2										
	>= 0.3	5	3	2	1	4	0	3										
	>= 0.1	0	2	4	2	2	2	2										
15_ Org	>= 0.5	0	0	0	0	0	0	0	0									
	>= 0.3	0	0	0	0	0	0	0	0									
	>= 0.1	0	0	0	0	0	0	0	0									
17_ Postcd	>= 0.5	4	4	3	5	4	5	2	5	0								
	>= 0.3	1	0	2	1	2	0	4	0	0								
	>= 0.1	1	2	0	0	0	1	0	1	0								
186_ Version	>= 0.5	0	0	0	0	0	0	0	0	0	0							
	>= 0.3	0	0	0	0	0	0	0	0	0	0							
	>= 0.1	0	0	0	1	1	0	0	0	0	0							
187_ Cipher	>= 0.5	0	0	0	0	0	0	0	0	0	0	0						
	>= 0.3	1	0	0	2	0	0	0	1	0	0	0						
	>= 0.1	3	0	1	2	3	0	4	1	0	0	1						
189_ CertVersion	>= 0.5	0	0	0	1	0	0	0	0	0	0	0	0					
	>= 0.3	0	0	0	0	0	0	0	1	0	0	0	0					
	>= 0.1	0	0	0	0	0	0	1	1	0	0	0	0					
244_ Cert	>= 0.5	1	0	0	2	1	0	1	1	0	0	0	0	0				
	>= 0.3	1	0	0	0	0	0	2	3	0	0	0	0	0				
	>= 0.1	2	0	2	1	2	0	3	1	0	0	0	3	0				
247_ NotVld	>= 0.5	1	0	0	0	0	0	0	1	0	0	0	0	0	0			
	>= 0.3	1	0	0	5	2	0	1	3	0	0	0	1	0	1			
	>= 0.1	3	0	1	2	4	0	4	2	0	0	1	3	1	3			
248_ After	>= 0.5	1	0	0	2	1	0	1	1	0	0	0	0	0	0	7		
	>= 0.3	1	0	0	0	1	0	1	2	0	0	0	0	0	1	0		
	>= 0.1	4	0	1	1	1	0	5	2	0	0	0	2	0	1	0		
250_ KeyLen	>= 0.5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	
	>= 0.3	0	0	0	1	1	0	0	1	0	0	0	0	0	1	0	1	
	>= 0.1	3	0	3	5	5	0	2	1	0	0	0	3	2	2	3	1	
288_ Version	>= 0.5	0	0	0	1	1	0	0	0	0	0	0	4	0	0	0	0	0
	>= 0.3	2	0	1	2	2	0	1	0	0	0	0	1	0	0	1	0	0
	>= 0.1	1	0	0	2	2	0	2	4	0	0	1	1	0	2	1	1	4

EFFECT SIZE	
Perfect	1.0
Large	>= .5
Medium	>= .3
Small	>= .1
Null Data	-

TABLE 21. BOTNET SAMPLES USED IN THE CREATION OF THE SSL CORRELATION MATRIX

Sample	Sample Date	Bot (VirusTotal)	SSL Flows
CTU25_1	09/09/2013	Zbot	20,664
CTU110_4	09/04/2015	HTbot	78,966
CTU111_2	09/04/2015	Unknown	42,562
CTU140_1	23/10/2015	Bunitu	64,816
CTU140_2	23/10/2015	Bunitu	46,647
CTU141_1	28/09/2015	Bunitu	74,160
CTU141_2	23/10/2015	Bunitu	81,691

4.6 Presenting the BotProbe IPFIX Templates

4.6.1 THE BOTPROBE TEMPLATE

The BotProbe template is shown in Figure 13. The template created a 43 byte PDU compared with the 48 byte NetFlow v5 PDU, where unlike the NetFlow template, each IE within this IPFIX template has been justified as a potential bot traffic attribute, with no superfluous fields. The IEs utilised within the template confine data export to attributes contained within the packet header, so the template can be used in environments that need to maintain payload privacy. Figure 13 forms the template used in BotStack (Chapter 5).

The BotProbe template flow record (Eq. 1) is the union of the static flow tuple (F_{static}) with the dynamic botnet characteristics tuple ($F_{dynamic}$):

$$F_{botprobe} = F_{static} \cup F_{dynamic} \quad \text{(Eq. 1)}$$

Where the tuples are defined as:

F_{static} = (***sIP, dIP, sPort, dPort, protocol***)

$F_{dynamic}$ = (***sTimeMS, eTimeMS, packets, iFlags, tcpSeq, collector***)

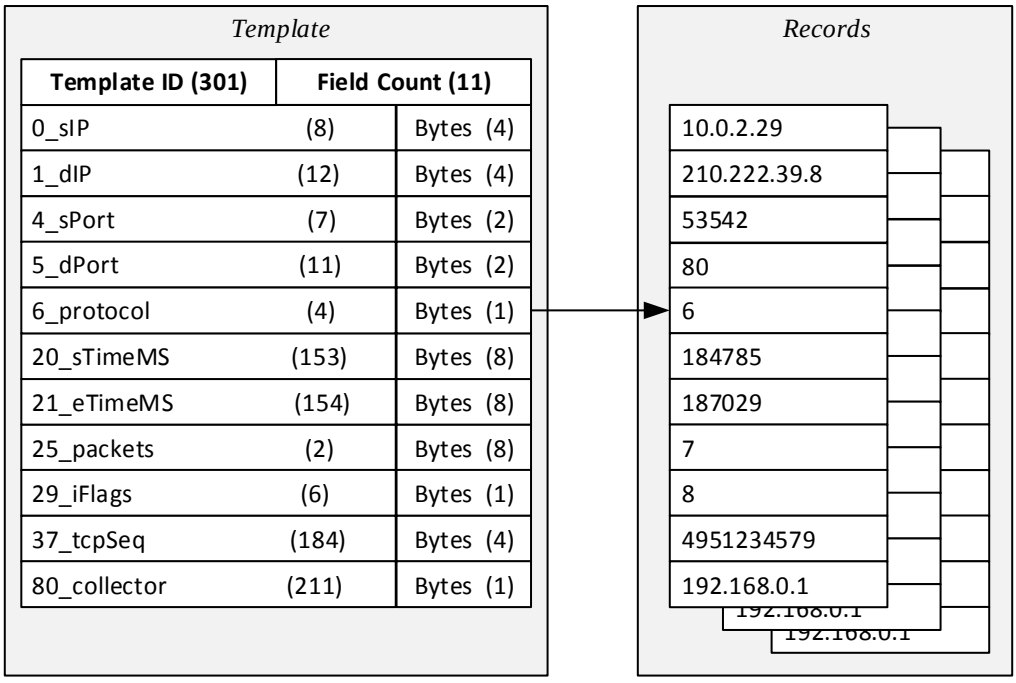


Figure 13. The BotProbe IPFIX template, comprising of 11 SuperMediator IEs with IANA ID# in brackets.

4.6.2 THE EXTENDED BOTPROBE TEMPLATE

The extended BotProbe template is shown in Figure 14. The addition of EEs into the template extends data export to application layer attributes, so the template can be used in environments when payload privacy is less sensitive. Figure 14 forms the template used in BotStack (Chapter 5).

The extended BotProbe template flow record (Eq. 2) is the union of the 11 field BotProbe template ($F_{botprobe}$) with the botnet protocol specific traffic tuples:

$$F_{extended} = F_{botprobe} \cup F_{identifier} \cup F_{HTTP} \cup F_{DNS} \cup F_{SMTP} \cup F_{SSL} \cup F_{IRC} \quad \text{(Eq. 2)}$$

Where the protocol tuples are defined as:

$F_{identifer}$ = (***flowKeyHash***)

F_{HTTP} = (***httpGet, httpResponse***)

F_{DNS} = (***dnsARecord, dnsSOARecord***)

F_{SMTP} = (***smtpHello***)

F_{SSL} = (***sslName***)

F_{IRC} = (***ircTextMessage***)

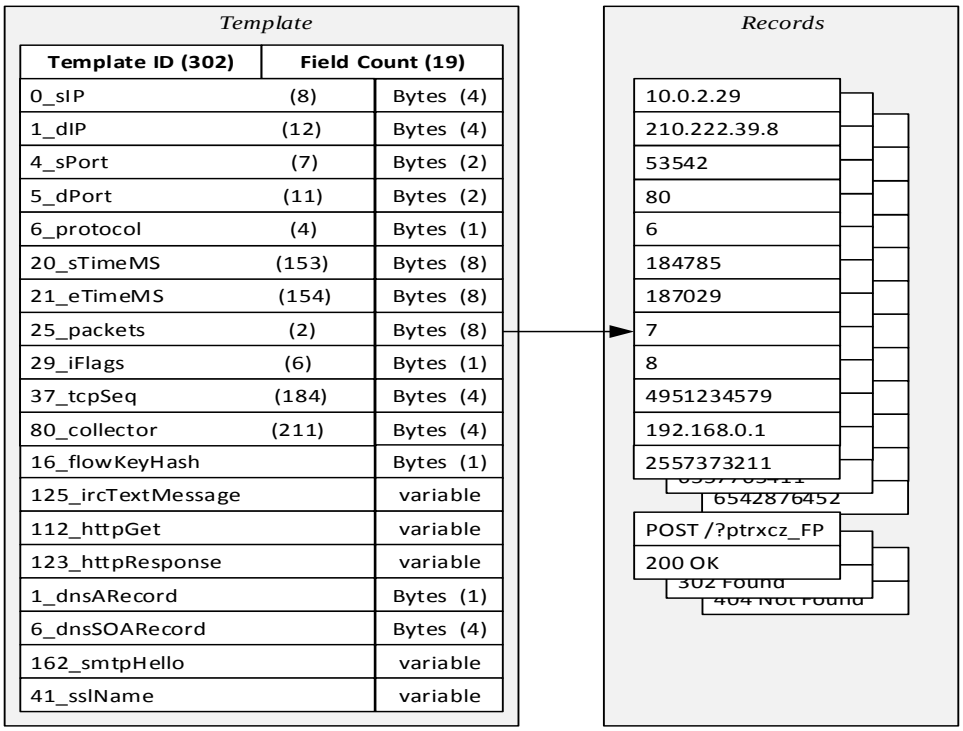


Figure 14. The extended BotProbe IPFIX template, including an additional seven EEs with IANA ID# in brackets.

4.7 Discussion of the BotProbe IPFIX Templates

Behavioural-based detection functions, such as botnet traffic detection, can quickly capture a lot of traffic. Multiple distributed probes across a diverse high-speed data network capture Gigabits of PCAP data per second per collector. Whilst this raises concerns for data storage, a larger impact is the overload of data that requires analysis, either manually or by machine. The challenge is in keeping the capture datasets to a manageable size for both storage and analysis, whilst capturing sufficient attributes to be able to identify known and as of yet unknown malware. Whilst flow export is a more focused data collection method compared to PCAP, each element in a template will impact both the volume of collected data as well as the time taken to process the flow. With flow export, if an element is present in a template but the flow contains none of this data, the field is still fully populated with null data. This is common to all flow exporters, not just YAF; although when YAF collects EE, field sizes are 0 bytes if the data is not present in the flow. Therefore each element within an IPFIX template requires justification for inclusion, to ensure full utilisation of the IPFIX PDU, unlike NetFlow.

4.7.1 A DISCUSSION OF INFORMATION ELEMENTS

A) SCATTER PLOTS

Figure 8 shows scatter plots for various IEs, from two different bot samples CTU3_1(Kelihos) and CTU25_1(Zbot). These two bot samples were unbiasedly selected at random from all samples being analysed (see Table 7) to demonstrate IE behaviour similarities and contrasts in different botnets.

When comparing TCP IEs (***iFlags, uFlags, rIFlags, rUFlags)***, the scatter plots look similar with all TCP IEs appearing with similar regularity. Likewise for packet and byte scatter plots (***packets, rPackets, bytes, rBytes)*** similar behaviour is presented between each IE within the sample. Similar behaviour is also shown with corresponding IEs across the two samples. First impressions of the packet and byte scatter plots are that most plots display two branches. This suggests two different correlation events within the IE, indicating that two different trends are occurring and that the data should be separated for further analysis. However, zooming in closer into the scatter plot shows that data forms a funnel shape as x-axis = 0, y-axis = 0 is approached. Hence, these branches actually indicate the limits of the data points, with data points only falling between the branches. The closer the branches, the closer the linear relationship, as can be seen with CTU3_1(Kelihos) ***packets*** and ***rPackets***

which closely resembles a linear relationship. This is confirmed by the strong correlation $r_s(238,841) = +0.994$, $\rho<0.001$, where r_s is Spearman's rho correlation, ρ indicates the probability of the correlation through chance and n is the total number of flows correlated in that sample.

The largest contrasts in scatter plot shapes are seen in the IP and port IEs. For example, differences are observed between the ***dIP*** and ***sPort***. In CTU3_1(Kelihos) this appears as a solid block indicating a wide range of ports and IPs, suggesting a SPAM attack or port scan. In CTU25_1(Zbot) the scatter plot suggests a more focused attack. Likewise for ***sIP*** and ***dPort***. When transformed values of ***sIP*** are correlated with ***dIP*** the L-shape scatter plot suggests high correlation between lower-range IP addresses. Closer inspection of the bot sample PCAP showed that most of the sample data was traffic to and from IP 8.8.4.4 and 8.8.8.8 hinting at DNS requests, confirming the suspicion of SPAM.

Figure 8 only shows a small selection of IE correlations. All IEs demonstrated variability in scatter plot shapes, however the IEs in Figure 8 where selected because they showed more "interesting" correlations. Figure 8 shows comparison between CTU3_1(Kelihos) and CTU25_1(Zbot), however similar variability in IEs was evident in all other samples.

B) FIVE FIELD FLOW TUPLE

The *seven field flow tuple* for categorising a unique flow is defined by Cisco as source IPv4, destination IPv4, input interface, source port, destination port, layer 3 protocol and type of service (*Patterson, 2012*). However, in security analysis this is often reduced to a five field tuple; dropping 'input interface' and 'type of service' (ToS) (*Santos, 2016*). The ***sIP***, ***dIP***, ***sPort***, ***dPort*** and ***protocol*** IEs all demonstrated 100% template field occupancy in testing. The medium to small association of ***sPort*** and ***dPort*** with both each other and with ***sIP*** and ***dIP*** (refer to Table 6) suggests that the traffic captured by these four IEs is suitably different to warrant all four IEs in the template, as these field are complementary rather than duplicating captured traffic. Conversely, ***protocol*** and ***dPort*** demonstrated a large association indicating the potential for duplication of traffic captured. This is expected, because in telecommunications a packet's destination port is usually set in accordance with the communication protocol used. What was unexpected, was that this correlation was not larger. A possible reason may be that bots utilise spoofing techniques as a defence

mechanism; assigning the protocol to different port numbers from those defined in the IANA port list.

The implication from this study that all five IEs are necessary in the BotProbe template is in agreement with other botnet detection research. The extent of the use of the five field tuple is evident in Table 2, above. However, several researchers have moved away from this five field flow tuple, choosing to drop either ***sPort*** or ***dPort*** (*Haddadi et al., 2014; Narang, Reddy and Hota, 2014; Lin, Chen and Chang, 2014; Zhang, et al., 2014; Françios, et al., 2011; Strayer, et al., 2008*). A reason for dropping ***dPort*** is port spoofing, whilst ***protocol*** gives a more accurate traffic description (*Dietrich, 2013*). However, some researchers chose instead to drop ***protocol*** from their algorithms (*Narang, et al., 2014; Françios, et al., 2011; Goeble and Holz, 2007; Gu, et al., 2008; Karasaradis, Rexroad and Hoeflin, 2007*). Whilst dropping any of the five fields from the tuple reduces template space, their research methodologies provide no evidence that this is a benefit. Occupancy for ***ToS*** (SuperMediator_ID#75) was 0.3%. This meant that over the 7,363,251 flows analysed, the ***ToS*** field used 7.16M bytes to capture just 19,583 bytes of data. With such a low occupancy and with no academic evidence as a useful botnet indicator, ***ToS*** was uneconomical to capture. Likewise for ***rToS*** (0.0% occupancy). This is in agreement with Gates, who suggested the type of service field is redundant in NetFlow v5 bot traffic capture (*Gates, et al., 2004*).

C) FLOW CONTEXTUAL IES

Several flow contextual flags demonstrated 100% occupancy; including ***eTimeMS***, ***sTimeMS***, ***duration*** and ***collector***. Over all 21 bot samples analysed, a perfect positive association was measured between ***sTimeMS*** and ***eTimeMS***: $r_s(7{,}363{,}251) = +1.00$, $\rho<0.001$, where the p value indicates the probability of this correlation through chance is < 0.1%. This large association is unexpected as each flow is assigned a start and end time. A large correlation effect size implies that two IEs are likely to be capturing similar data, either duplicating of reinforcing data capture. A perfect association would suggest that all flow durations are equal length, i.e. the difference between ***eTimeMS*** and ***sTimeMS*** was constant. Visual observation of the data showed that this perfect correlation is misleading. In actuality, the flow durations did fluctuate, but the duration times were small when compared to the start or end times. Therefore, during correlation calculations these difference approached 0, resulting in a perfect correction, when in fact the relationship is far from perfect, as is shown by the

duration IE. Both ***sTimeMS*** and ***eTimeMS*** are required to calculate the duration of a flow and to correctly order flows, so as to map malware propagation across a network in real time. Rather than calculate duration from two 8 byte fields, a more template space efficient method may be to capture ***duration*** (SuperMediator_ID#17) as a 4 byte field. The ***duration*** IE showed a large association with ***packets*** and ***bytes***, and a medium association with the reverse equivalent IEs ***rPackets*** and ***rBytes***, suggesting duplication of captured data. The ***duration*** IE also showed a small correlation with ***sTimeMS*** and ***eTimeMS*** timing IEs. This is not unexpected, as the duration of a flow is not anticipated to correlate with the times the flows started, unless a bot called home for the same duration at the same time each day. Whilst it may be efficient to replace one of the timer IEs with ***duration***, both ***sTimeMS*** and ***eTimeMS*** timers were included in the template, as the accuracy of ***duration*** has yet to be fully tested. This is in agreement with Gates who retained flow timers for security analysis in NetFlow v5 (*Gates et al, 2007*) and most other botnet researchers (as summarised in Table 2, above). IPFIX supports flow timers at nanosecond granularity (IANA_ID#156 and IANA_ID#157). At present, neither YAF nor SuperMediator are able to support these IANA_IDs. Additionally, measurement to nanoseconds requires specialist network cards and other equipment, which may not be present in a cloud environment.

YAF assigns each flow a ***collector*** flag indicating the device from which the flow export originated. The data samples under test in this study were in PCAP format, hence YAF was unable to extract collector/exporter information as it is not present in PCAP, meaning that this IE could not be fully analysed. However, ***collector*** will be retained in the template to provide useful information as to where a bot is captured in the network and how it propagates. YAF uses ***attribute*** as a flow context flag, which is set to 1 when all forward direction flow packets are of a fixed size. A low field count for both ***attribute*** (12.1% occupancy) and ***rAttribute*** (4.7% occupancy) suggested these two attributes have little relevance to botnet traffic. Due to low occupancy, correlation figures could not be calculated for ***rAttribute***. Literature provided no evidence for the use of these two IEs in botnet detection, so both were discarded. The ***endReason*** flag can be set to *“”, active, idle, force, rsrc* or *eof* to indicate why a flow has terminated. As a contextual flag, ***endReason*** is unlikely to be influenced by botnet traffic. This flag showed a large association with ***iFlags*** and a medium association with ***protocol*** and other IEs, and therefore can be considered to be duplicating other information captured in IEs that might be more relevant.

D) PACKETS AND BYTES IES

IEs ***packets*** and ***bytes*** had 100% occupancy, whilst the occupancy of ***rPackets*** and ***rBytes*** was slightly lower. A large association was measured between ***packets*** and ***bytes***. Unexpectedly, this correlation was not measured as perfect, possibly because the samples contained a low number of small packets of only a few bytes which were recorded as zero packet size. A perfect positive association was measured between ***packets*** with ***rPackets***: $r_s(7{,}363{,}251) = +1.00$, $\rho<0.001$. The association between ***bytes*** and ***rBytes*** was not quite so large, possibly for the same reason of small bytes being recorded as zero size. As expected, packet and byte flows showed large associations with ***iFlags*** (which holds the initial TCP flag in a flow) and medium correlation with ports and protocol IEs. As these IEs capture similar traffic data, ***packets*** will be retained in the template as it shows larger correlation with other IEs. Gates made no distinction between whether packets or bytes is the best indicator (*Gates, et al., 2004*). Several researchers chose to capture both ***bytes*** and ***packets*** fields. Whilst others, in line with the results obtained during the creation of the BotProbe template, chose to drop ***bytes*** in favour of ***packets*** (*Lin, Chen and Chang, 2014; Zhao, et al., 2013; Rossow, et al., 2011*). It could be argued that bot keep-alive packets are so small, they may not register in ***packets*** and that ***bytes*** would make a better attribute. However, during conceptual validation in chapter 6, no evidence could be found that this was the case and Zeus was successfully witnessed sending keep-alives captured as ***packets***.

E) APPLICATION IES

IANA defines three IEs for capturing application information; elementID 94-applicationDescription (string), 95-applicationID (array) and 96-applicationName (string). YAF identifies the application description from the protocol and port numbers, and assigns a corresponding number into the ***application*** field. Effectively, ***protocol*** and ***application*** both capture the same data, so, as expected, ***protocol*** and ***application*** exhibited a large correlation association. A reason for this not being a perfect association is that ***application*** only retains a list of common applications rather than an exhaustive list of all applications. Hence the occupancy for ***application*** was lower at 59.5%, against 100% for ***protocol***. A large association was measured between ***application*** and TCP flag IEs ***iFlags*** and ***uFlags.*** However, a marginally larger association was measured between ***protocol*** and these same TCP flag IEs. Therefore, as the data showed ***application*** to have a lower field occupancy than ***protocol***, ***application*** was dropped in favour of ***protocol***.

F) TCP IEs

TCP flag IEs holds discrete handshake flags (SYN, ACK, FIN, RST, *et cetera*) according to TCP flow status. The IE ***iFlags*** holds flags for the initial packet in the flow, whilst ***uFlags*** holds flags for the subsequent packets. The ***tcpSeq*** IE contains the TCP number that is randomly assigned when the host initiates a TCP session. A large correlation was measured across all six TCP IEs. A medium sized correlation was also measured between ***protocol*** and ***packets***, and ***protocol*** and ***bytes***, suggesting that data is being duplicated. IEs ***protocol*** and ***packets*** have both already been confirmed above as being required in the template. Of the four TCP flags (***iFlags***, ***uFlags***, ***rIFlags*** and ***rUFlags***), the initial flags (***iFlags***) showed the largest correlation with the other TCP flags and likewise had the highest occupancy. This suggests ***iFlags*** was the most useful of these attribute in botnet traffic capture, as the initial TCP flow contains marginally more useful handshake information than subsequent flows. As such, only ***iFlags*** was retained in the template.

The low occupancy of the TCP flag IEs was expected to be because the samples contained both TCP and UDP traffic; whilst TCP flags were only assigned to TCP flows. Low occupancy may suggest that TCP flags add little value in botnet detection. However the Blaster worm was detected because multiple SYN flows with fewer ACK replies indicated port scanning activity (*Dübendorfer, et al., 2005*). Capture of initial TCP flags (***iFlags***), and union TCP flags (***uFlags***) are specific to IPFIX, which may explain why not much literature could be found for comparison studies. *Rincón, et al.*, (2015) suggested that both initial and union TCP flags can be utilised in the search for incomplete TCP flows in DNS traffic, but provided no empirical evidence to back this up. Despite a low occupancy and similar large correlation to the four types of TCP flag IEs, ***tcpSeq*** has also been retained in the template, as it can be used to group common flows during analysis.

G) REVERSE FLOWS

Six of the 23 IEs available in YAF/SuperMediator export reverse flow IEs. All measured either a large or medium correlation with their forward counterpart (excluding ***rAttribute*** for which no correlation data was generated, as explain above) suggesting little benefit in capturing the reverse IEs. Reverse IEs are not defined by IANA, which was another argument to exclude reverse IEs in the template, in order to maintain a standards compliant IPFIX template.

4.7.2 A DISCUSSION OF ENTERPRISE ELEMENTS

Many botnet detection algorithms utilise data at an application layer, as is evident in Table 2 above. An example of application data often used by a detection algorithm is HTTP GET, which can hold data specific to a botnet family type. Therefore, whilst data from a packet header (captured by IEs) can indicate the presence of malicious activity, application data (typically captured by EEs) can confirm traffic is of botnet origin (*Husák, Velan and Vykopal*, 2015). YAF treats EE export differently to IE export. When an EE is present in an IPFIX template, if the flow does not contain that EE data the export field is not populated. This is unlike IEs where the export field is fully populated with null data. This meant that the justification for inclusion of EEs in the template based on data export volume efficiency was less of a requirement than inclusion of IEs. However, due consideration was still applied to EEs as they typically export data in variable length strings, which can impact PDU size if an EE is included in an IPFIX template. There was also some expectation that EEs would impact IPFIX software processing time, regardless of whether data was exported or not.

The overall occupancy of the EEs tested was noticeably lower than the occupancy values of IEs, suggesting that EEs are less reliable botnet traffic attributes and that their export is inefficient. This argument can be countered by EEs not populating the IPFIX PDU if they are not present in the flow (as above). Of the 17 million EE flows analysed, two out of the 51 EEs had an occupancy >25% and 37 EEs had an occupancy of <10%. Low occupancy resulted in fewer data points during correlation. In some of the correlation matrices, correlation scores were not significant enough to be used, so are marked as *null data* in Tables 14, 16, 18 and 20. This meant only 23 bot samples contained EEs that could be used for study. In some instances, such as for SMTP analysis, a population of as few as four bot samples could be found in the CTU Prague repository. With fewer samples available for study, the results obtained from EE analysis should be considered to be less reliable than IE results, which had more samples. However, as mentioned above, the implications of this are lessened as the inclusion of EEs in the template has less of an impact on export data volumes.

A) SCATTER PLOTS

Figures 9 - 12 show scatter plots for key EEs across the protocols under test: HTTP, DNS, SMTP and SSL. A scatter plot for IRC was not created as IRC only had a single variable. As with the IE scatter plots, plots for different bot samples are shown in these figures for comparison. The bot samples used to create these four figures vary,

as no one sample contained all protocols. The EEs in these scatter plots were chosen as displaying the most "interesting" features, however all EEs correlated, including those not shown, demonstrated some degree of similarity across all samples.

In Figure 9, the ***length*** EE scatter plots show similar behaviour, as HTTP payload length was small for the majority of HTTP packets. There are stark contrasts in the other HTTP EEs. In CTU25_1(Zbot) ***get*** against ***host*** plots a T-shape, suggesting a lot of variable GET requests from a specific host, indicating a network scan. Compared with CTU145_1(uTorrent), the same EEs have clusters of data points, suggesting more direct HTTP traffic. More differences are seen in DNS (Figure 10). For CTU3_1(Kelihos), ***a*** against ***cname*** shows a very different plot to CTU149_1(Kelihos) which is a similar botnet attack. This is because the CTU149_1(Kelihos) sample contained fewer DNS CNAME-records. Similar can be seen with DNS ***a*** against ***aaaa***. Both plots show a definite semi-linear correlation, but CTU149_1(Kelihos) has fewer outliers as the sample had fewer DNS AAAA-records. If this sample was captured over a longer period, the plots would be expected to trend towards being more similar, as the samples were from similar botnets. This might indicate that more active samples provide more reliable analysis. Again, SMTP scatter plots (Figure 11) show similarities and differences. CTU3_1(Kelihos) had more variable SMTP_HELO packets compared to CTU149_1(Kelihos), which becomes more evident when comparing ***to*** and ***response*** in the two samples. The SSL scatter plots (Figure 11) probably show the most consistent behaviour between bot samples, indicating that SSL has a similar functionality across all bots.

All EE scatter plots demonstrated high variability. This is confirmed by their correlation coefficients, which are mostly medium to small. Less linearity indicates that EE behaviour varies between bot family. More work is needed to understand if patterns between EEs can be used as signatures to detect specific bot families.

When correlating EEs, bot samples with fewer flows were found to be less reliable. Therefore, for HTTP and DNS, flows of >100,000 were analysed. For SSL and SMTP flows >10,000 were analysed, because only smaller flow sample sizes were available for study.

B) HYPERTEXT TRANSFER PROTOCOL EES

HTTP (TCP port 80) is the default application layer protocol for traffic transmission over the Internet. HTTP is also a popular bot C&C channel due to the difficulty in blocking port 80. Of the 30 bot samples that contained HTTP traffic, 17 samples were

over the 100,000 minimum flow threshold required for analysis, as outlined in the results analysis above. Of these 17 samples, low field occupancy in 10 samples resulted in insufficient data points to produce statistically significant correlation scores. No evidence was found for correlation association between the available HTTP EEs and YAF IEs.

YAF included 17 HTTP EEs for analysis. The majority of these exhibited small correlation effects with each other, indicating little duplication in captured data between HTTP EEs. This suggested that whilst occupancy was low, these EE are meaningful attributes to capture in bot traffic. Of the 17 HTTP EEs, only four EEs exhibited an occupancy above 10%, casting into doubt the usefulness of the remaining 13 EEs with occupancy below 10%. No academic evidence could be found to support that ***httpVersion, httpLocation, httpContentLength, httpAge, httpAccept, httpContentType, httpLanguage, httpSetCookie, httpAuthorization, httpVia, httpXForward*** or ***httpRefresh*** are indicative of botnet traffic attributes. Visual inspection of the captured data provided no further evidence on retaining these EEs, as they containing data of little use in detection. For example, ***httpLength*** captures the length of the GET request, which can be easily calculated if required.

Only two EEs demonstrated occupancy and valid content: ***httpResponse*** (SuperMediator_ID#123) (n=13.2%) holds response fields parameters and ***httpGet*** (SuperMediator_ID#112) (n=13.3%) holds GET/POST URIs. Both EEs exhibited a medium to large correlation effect size with other EEs that have been used to botnet detection in previous studies, such as ***httpUserAgent*** which was used to fingerprint browser characteristics in the detection of the Storm P2P bot (*Holz, et al., 2008*). Additionally ***httpGet*** holds domain or IP address, request parameters and cookies duplicated in other EEs. This is in agreement with BotSniffer which was amongst the first engine to feed HTTP GET/POST information into their algorithm (*Gu, Zhang and Lee, 2008*). From a network security point of view, four HTTP fields are *critical*: ***httpUserAgent***, ***httpReferer***, ***httpHost*** and ***httpCookie*** (*Collins, 2014)*. Within this research project, no evidence could be found to justify capture of these four EEs, but ***httpGet*** and ***httpResponse*** exhibited a large correlation with ***httpUserAgent***, ***httpHost*** and ***httpCookie***. *Husák, Velan and Vykopal* (2015) selected seven HTTP attributes to identify bots, including ***httpUserAgent***, ***httpReferer***, ***httpHost***, ***httpContentType***, ***httpResponse***, ***httpPath*** and ***httpRequestMethod***. Whilst ***httpPath*** and ***httpRequestMethod*** are available as EEs in YAF, ***httpGet*** contains similar information.

C) DOMAIN NAME SYSTEM EES

DNS **(**TCP/UDP port 53) converts domain names to IP addresses. DNS record attributes are commonly used by ISPs for botnet takedown, as bots use DNS to obtain the current IP address of their C&C servers. However, it was not until 2016 that a bot called Pisloader, was found to use DNS as a C&C channel (*Chickowski, 2016*). Five of the 15 bot samples analysed produced reliable correlation scores. No samples had occupancy sufficent to calculate correlation effect sizes for ***dnsMXRecord*** or ***dnsTXTRecord***. No evidence was found for correlation association between the available DNS EEs and YAF IEs. The largest occupancies were measured in ***dnsARecord*** (53.8%) and ***dnsSOARecord*** (18.2%). Visual inspection of the captured data showed that these two EEs contained data that may be useful for botnet traffic identification: ***dnsARecord*** (SuperMediator_ID#1) contained the host domain or IPv4 address giving a potential indication of the C&C server, and ***dnsSOARecord*** (SuperMediator_ID#6) marked the start of the authority zone, which was highly populated in the SPAM bot samples such as CTU25_5(Zbot) (although this sample failed to produce reliable correlation figures so was excluded from analysis). Perfect association was measured between ***dnsARecord*** and ***dnsAAAARecord*** which holds the IPv6 equivalent to the IPv4 address held in ***dnsARecord***. A large association was also measured between ***dnsARecord*** and ***dnsSRVRecord*** and medium correlation with ***dnsCNAMERecord***, reinforcing its inclusion in the template.

D) SIMPLE MAIL TRANSFER PROTOCOL EES

SMTP (TCP ports 25/587) is an email transport protocol. Other proprietary email protocols exist but are not supported in YAF/Supermediator. Four bot samples provided reliable correlation scores. No evidence was found for correlation association between the available SMTP EEs and YAF IEs. Visual inspection of the captured data revealed only ***smtpHello*** (SuperMediator_ID#162) to contain useful information; holding both the SMTP command (HELO, AUTH, FROM, *et cetera*), and more importantly holding the sender IP address or domain which could be used to traceback the originator. This could be more reliable than ***smtpFrom*** which can be spoofed. IEs ***smtpTo*** and ***smtpRresponse*** exhibited a medium association, but both contained victim data rather than originator. The ***smtpSubject*** EE has potential to export SPAM when it contains nefarious words in the subject title. However, with a low occupancy, no academic evidence could be found to suggest this EE was worth retaining in the template.

E) SECURE SOCKET LAYER EES

SSL (TCP port 443 for HTTPS and several other ports) provides encrypted Internet communications. No academic evidence could be found suggsting that bots currently use SSL as a communication channel. Even so, SSL still comprised over 2.6% of the bot sample traffic, indicating that bots may use SSL to encrypt communication. Seven of the 16 bot samples analysed produced reliable correlation scores. None of the SSL EEs correlated with IEs. Overall the SSL EEs had a low occupancy with no EE above 15%, but many correlations between EEs were evident. Visual inspection revealed that overall the SSL EEs contained little useful information. Perhaps the most useful being ***sslName*** (SuperMediator_ID#41) which contained certificate authority details, including some interesting Russian domains and IP addresses, which could possibly be used to trace the originator. The ***sslName*** EE has been included in the template, but little acadmic evidence could be found to suggest inclusion of other SSL EEs.

F) INTERNET RELAY CHAT EES

IRC (TCP ports 6600-6669 and 7000) is an application layer protocol for client/server text chat over the Internet. IRC was the original botnet communication channel, and is still in common use. IRC is much studied as an indicator of botnet traffic. IRC header and IRC channel are commonly used attributes (*Gu, Zhang and Lee, 2008; Karasaradis, Rexroad and Hoeflin, 2007; Goeble and Holz, 2007; Gu, et al., 2007*). YAF only supports one IRC field, ***ircTextMessage*** (SuperMediator_ID#125), which captures USER, JOIN, NICK, USERHOST and associated chat. The EE ***ircTextMessage*** has been retained in the template capturing IRC header information. IRC channel information can be captured by ***sPort*** and ***dPort***. Retention of both of these attributes is in agreement with previous academic studies.

4.7.3 THE BOTPROBE TEMPLATE VERSUS THE EXTENDED BOTPROBE TEMPLATE

CSP tenants have an expectation that their data is not under surveillance by the CSP, for anomaly detection or otherwise, with privacy requirements enforced within a CSP environment. IEs export data contained within a packet header, which can be considered to be in the public domain, particularly in the Internet. EEs export data which tends to be transmitted within the packet payload, which is traditionally a private domain. Anecdotal evidence from conversations with cloud providers during this study suggests that some CSPs are being approached by tenants who are willing to

trade a degree of access to payload data in return for a highly level of security. This prompted the creation of the extended BotProbe template, to demonstrate that the technology exists to support EE capture, even if privacy rules prohibit their capture. The BotProbe template has been designed to use IEs defined by IANA to ensure vendor interoperability. This means that the BotProbe template is not restricted to use only with YAF, and should be reproducible on any IPFIX exporter/collector pair. EEs are not defined by IANA, but are bespoke to a vendor. Therefore, the extended BotProbe template is less likely to work outside of a YAF environment. At present, the only other IPFIX exporter with support for EEs is nProbe. The extended BotProbe template could be replicated in nProbe by replacing %HTTP_URL with ***httpGet,*** %HTTP_RET_CODE with ***httpResponse,*** %DNS_QUERY with ***dnsARecord*** and %SMTP_MAIL_FROM with ***smtpHello***. NProbe does not have an equivalent for ***dnsSOARecord, ircTextMessage*** or ***sslName***. Whilst these nProbe EEs are similar to YAF, there is no guarantee they will capture identical information. The extended BotProbe template also includes ***FlowKeyHash*** (SuperMediator_ID#16) in the IE `FIELDS`. SuperMediator outputs EE traffic into separate `.txt` files, defined in SuperMediator `DPI_CONFIG TABLES`. ***flowKeyHash*** is a cross reference field between EE flows in the `.txt` file and IEs exported in "`flow_records.csv`". This field is not necessary when using nProbe, as nProbe outputs both IEs and EEs to the same file.

4.7.4 IMPLICATIONS OF CAUSE AND EFFECT

It should be noted that correlation does not imply effect (*Clegg, 1995*). Correlation is a measure of association between two variables, i.e. how likely two elements are to be performing similar roles in the template. Causality of independent variables (cause) upon a dependent variable (effect) is often difficult to prove with statistics and instead is proven via direct experimentation. Nor does regression imply cause and effect (*Clegg, 1995*). Regression is used to indicate how the relationship between a set of independent variables can predict an outcome. In this study, field occupancy is used as an indicator that a variable is present in sufficient quantities in botnet traffic to warrant capture, whilst correlation is used to see if two or more elements are duplicate data being captured. It is wrong to suggest that because a variable such as ***sIP*** occurs in this traffic (cause) that there must be a botnet present (effect), as ***sIP*** is common to all traffic not just botnets. Similarly, it is wrong to use regression to suggest protocol number cannot be directly calculated from ***sIP*** addresses, even though they have a large association. Determining cause and effect is the role of the detection algorithm, which is beyond the scope of this study.

4.8 BotProbe Performance Test Methodology

Despite *Gates, et al.,* (2004) suggesting that NetFlow v5 comprises data fields that are not of use in security analysis, NetFlow v5 has still continued to be used in the academic study of botnet traffic detection. Table 2, above, summarises the attributes used by previous botnet detection algorithms; indicating that shortfalls in the attributes available in NetFlow v5 are supported by using PCAP.

This research project presents the hypothesis that IPFIX offers advantages over NetFlow v5. This chapter has already demonstrated how template extensibility allows the creation of more focused botnet detection templates than that of NetFlow v5. The data collection for each of the studies in Table 2 can be undertaken using either the BotProbe template as a direct replacement for capture of the NetFlow v5 elements, or the extended BotProbe template as a direct replacement for the capture of the NetFlow v5 elements and the additional elements captured in PCAP (*Zhang, et al., 2014; Rossow, et al., 2011; Yen and Ritter, 2010; Gu et al., 2008)*. Template extensibility has allowed the novel BotProbe templates to perform the data capture more efficiently, as the templates are no longer fixed at 48 bytes like in NetFlow v5. For example, the nine fields captured by *Gates, et al.,* (2004) totalled 30 bytes, but still uses the full 48 byte PDU to capture these nine fields. *Wijesignhe, Tupakula and Varadharajan* (2015) claim to capture their attributes in a 30 byte IPFIX PDU. The results they provide however, indicate that their study used NetFlow v9, not IPFIX, so were unable to capture EEs.

The remainder of this chapter provides empirical evidence of the performance of the two BotProbe IPFIX templates against NetFlow v5. In order to further satisfy the hypothesis that IPFIX offers advantages over NetFlow v5, performance comparisons will be made in:

(1) Template processing times;

(2) Overall data volumes captured by the templates;

(3) The impact of each template upon the device CPU loading.

4.8.1 CREATING A NETFLOW V5 TEMPLATE FOR BENCHMARKING

YAF is solely an IPFIX exporter, with no support for NetFlow v5 export. In order to gain comparative performance measurements, the testing of IPFIX against NetFlow needed to be undertaken on the same device. This required the creation of a NetFlow v5 template in IPFIX, so that NetFlow v5 export can be simulated by YAF. Figure 15 provides a side-by-side comparison of an actual NetFlow v5 PDU (left) with the IPFIX simulated PDU (right), showing the SuperMediator ID field numbers. It is not possible to create an exact replica of NetFlow v5 in IPFIX, as the NetFlow v5 fields do not map directly into IANA IPFIX IEs; with the exception of srcAddr, dstAddr, srcPort, dstPort, tos and proto which do map directly. As the performance tests were to validate the speed and volumes of capture data, rather than field content related, the remaining twelve NetFlow fields were simulated using fields of identical size rather than content. This meant that both the actual NetFlow template and the simulated template were 48 bytes in length. IPFIX does have equivalent IEs to NetFlow's dPkts, dOctets and first and last fields, however, these cannot be directly mapped, as IPFIX future proofs these fields with support for IPv6 at 8 bytes, versus 4 bytes in NetFlow.

NetFlow v5 Template

NetFlow v5	Field Count (20)
srcAddr	Bytes (4)
dstAddr	Bytes (4)
nextHop	Bytes (4)
input	Bytes (2)
output	Bytes (2)
dPkts	Bytes (4)
dOctets	Bytes (4)
first	Bytes (4)
last	Bytes (4)
srcPort	Bytes (2)
dstPort	Bytes (2)
padding	Bytes (1)
tcpFlags	Bytes (1)
proto	Bytes (1)
tos	Bytes (1)
srcAS	Bytes (2)
dstAS	Bytes (2)
srcMask	Bytes (1)
dstMask	Bytes (1)
padding	Bytes (2)

Simulated NetFlow v5 Template

Template ID (303)		Field Count (20)
sIP	0	Bytes (4)
dIP	1	Bytes (4)
domain	13	Bytes (4)
vlan	15	Bytes (2)
application	7	Bytes (2)
tcpSeq	37	Bytes (4)
rTcpSeq	38	Bytes (4)
ingress	52	Bytes (4)
egress	53	Bytes (4)
sPort	4	Bytes (2)
dPort	5	Bytes (2)
iFlags	29	Bytes (1)
rIFlags	30	Bytes (1)
protocol	6	Bytes (1)
ToS	75	Bytes (1)
attribute	33	Bytes (2)
rAttribute	34	Bytes (2)
uFlags	31	Bytes (1)
rUFlags	32	Bytes (1)
firstNonEmpty	81	Bytes (2)

Figure 15. A comparison of the NetFlow v5 template with the template simulated in IPFIX Actual NetFlow v5 Template (left); simulated NetFlow v5 template (right)

4.8.2 DATASET

As with the template creation tests, botnet samples used in the performance tests were adopted from the malware repository at CTU University, Prague.

4.8.3 EQUIPMENT

The test environment was identical to the environment used in the creation of the IPFIX templates, above.

4.8.4 TEST #1 - PROCESSING TIME TESTING METHOD

The aim of this test was to quantify the differences in execution times between (1) the BotProbe template and NetFlow v5 and (2) the extended BotProbe template and NetFlow v5. The independent variables were the three templates; BotProbe, extended BotProbe and NetFlow v5. The dependent variables were 10 randomly selected botnet samples (refer to Table 22), chosen to ensure a variety of botnet families were tested. Flow diagram Figure 16 summarises the testing method.

The detailed processing time testing method was:

(1) Before testing began, any Ubuntu system processes running in the guest VM that were unnecessary to the data capture test were disabled;

(2) A bot sample was selected, at random, from the CTU repository;

(3) YAF was configured to convert the `.pcap` sample into `.yaf` IPFIX format:

```
# yaf --in in_file.pcap --out out_file.yaf -v
--plugin-name=/usr/local/lib/yaf/dpacketplugin.la
--applabel --max-payload 65535
```

(4) A python script was executed against the appropriate template under test:

```
# python timer.py botprobe.conf
# python timer.py extended.conf
# python timer.py nfv5.conf
```

(5) The python script started a timer, twice executed SuperMediator with the appropriate test template, stopped the timer and calculated the total time taken;

(6) The python script was executed a total of 10 times, to minimise the noise effect from background processes;

(7) The fastest six run times were recorded to calculate a mean score;

(8) The test was repeated for all three templates.

SuperMediator templates can be found in Appendix C.

The python script can be found in Appendix D.

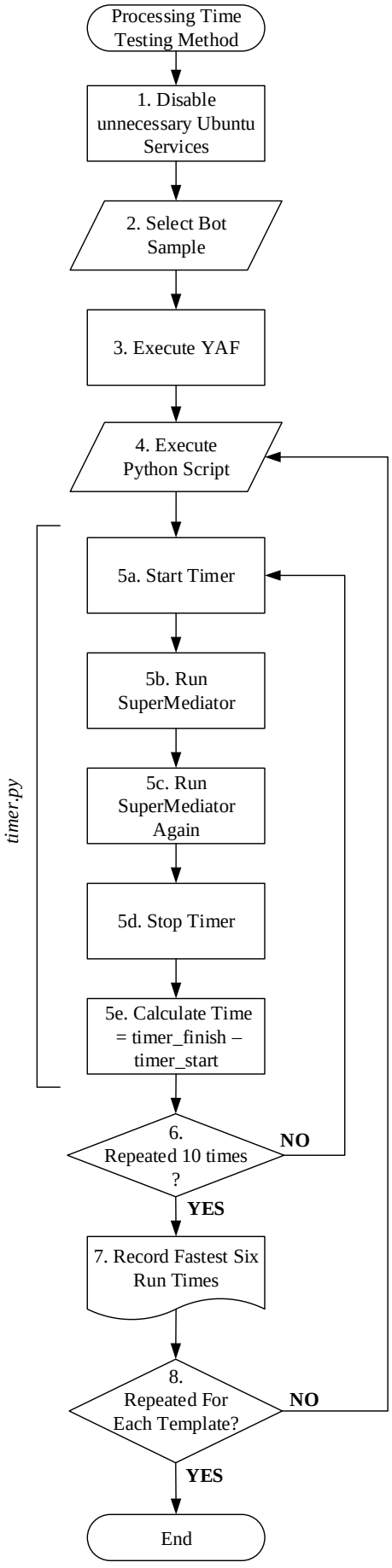


Figure 16. Flow diagram of the processing time test.

4.8.5 TEST #1 - PROCESSING TIME TESTING ANALYSIS

To quantify the differences in processing times between the IPFIX templates, the mean execution time was compared for each template. To minimise the noise effect from background Ubuntu processes upon the CPU, each template was executed 10 times with the slowest four times rejected, as explained below. The mean of the six fastest runs allows experimental error values to be calculated.

The processing time of each template was measured using a python script (Appendix D) which calculated the time taken for SuperMediator to export the `.yaf` file to `.csv`. The python script executed SuperMediator twice and displayed the quickest time from the two iterations. During the experiment design phase it was found that regardless of how many times the python script consecutively executed a batch of SuperMediator captures, in approximately 70% of cases the first SuperMediator run was the quickest. Times of subesquent runs in the batch varied with no apparent pattern. This was contrary to expectation as the first run of SuperMediator in a batch creates any new output files needed, with subsequent runs overwriting these. It would be expected that creation of new files would take longer than overwriting existing files. The reason for this was unknown. It was anticipated that there may be some lag due to SuperMediator terminating processes between subsequent runs. To account for this irregularlity the python script need only run SuperMediator twice in each test batch, with the quickest of each run recorded. There was no advantage gained from running large batches of SuperMediator.

It was also found during the design phase that background processes in the Ubuntu testing VM created noise that varied the times taken to run the processing tests. It was considered unfeasible to eliminate all background noise, as this involved stopping processes that were required during test. In most instances, the operating system automatically restarted these stopped processes. The following steps were taken to minimise background noise:

- Prior to each test session, all unnecessary services were disabled in the testing VM, after which the testing VM was given a two minute period to allow all processes to stabilise;
- Neither the testing VM nor the host device were touched whilst a test was running, so as not to initiate any new processes;
- Each test was run 10 times with the fastest six (found to be the optimum number during the design phase) results used to create a mean score.

4.8.6 TEST #2 - DATA VOLUME TESTING METHOD

The aim of this test was to quantify the differences in data volumes captured between:

(1) The BotProbe template and NetFlow v5;

(2) The extended BotProbe template and NetFlow v5;

(3) The BotProbe template and PCAP.

The independent variables were the three templates. The dependent variables were 10 randomly selected botnet samples (refer to Table 23), chosen to ensure a variety of botnet families. Flow diagram Figure 17 summarises the testing method.

The detailed data volume testing method was:

(1) Before testing began, any existing `.yaf` or SuperMediator (`.csv`, `.txt`) created files were deleted, so as not to cause confusion with new output files;

(2) A bot sample was selected, at random, from the CTU repository;

(3) The size of the bot sample `.pcap` file was recorded;

(4) YAF was configured to convert the `.pcap` sample into `.yaf` IPFIX format:

```
# yaf --in in_file.pcap --out out_file.yaf -v
--plugin-name=/usr/local/lib/yaf/dpacketplugin.la
--applabel --max-payload 65535
```

(5) The IPFIX stream ("`out_file.yaf`") was fed into SuperMediator, which exported the appropriate template under test:

```
# super_mediator --config botprobe.conf
# super_mediator --config extended.conf
# super_mediator --config nfv5.conf
```

(6) The sizes of the output files were recorded.

(7) The test was repeated for all three templates.

SuperMediator templates can be found in Appendix C.

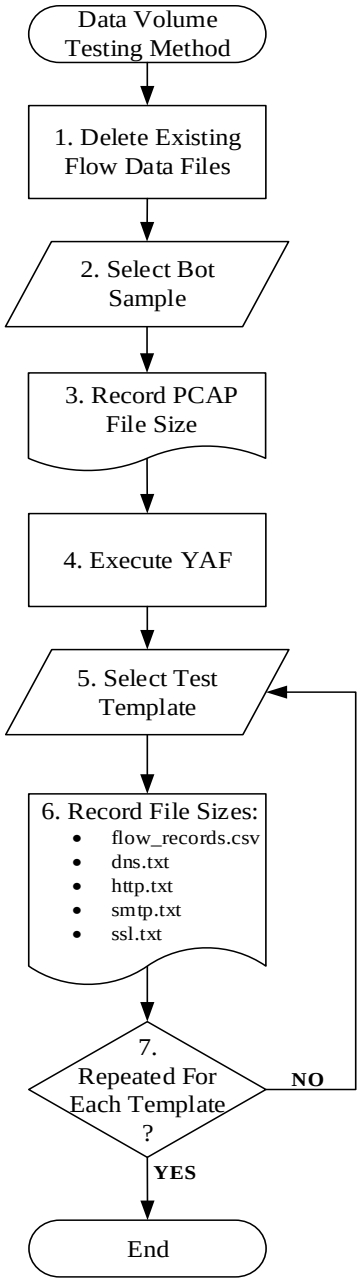


Figure 17. Flow diagram of the data volume test.

4.8.7 TEST #2 - DATA VOLUME TESTING ANALYSIS

To quantify the differences in data volumes created by the templates, the size of the output files were directly compared for each template. During the experiment design phase, it was found that when a test was rerun without any changes to the dependent or independent variables, each run gave identical data volume measurements. This was unlike the processing time test and CPU load tests, in which background Ubuntu system processes impacted speed measurements as undesireable noise. Therefore, each volume test was only performed once, giving no experimental error figures from test noise.

4.8.8 TEST #3 - CPU LOAD TESTING METHOD

The aim of this test was to quantify the difference in impact upon the test device CPU loadings from each template. The independent variables were the three templates. The dependent variables were four randomly selected botnet samples. Flow diagram Figure 18 summarises the testing method.

The detailed CPU load testing method was:

(1) Before testing began, unnecessary services in the guest VM were disabled;

(2) A bot sample was selected, at random, from the CTU repository;

(3) YAF was configured to convert the `.pcap` sample into `.yaf` IPFIX format:

```
# yaf --in in_file.pcap --out out_file.yaf -v
--plugin-name=/usr/local/lib/yaf/dpacketplugin.la
--applabel --max-payload 65535
```

(4) The IPFIX stream ("`out_file.yaf`") was fed into SuperMediator, to export the appropriate test template. SuperMediator was not executed at this stage:

```
# super_mediator --config botprobe.conf
# super_mediator --config extended.conf
# super_mediator --config nfv5.conf
```

In a separate terminal window, a python script was executed:

```
# python cpu_load.py
```

(5a) A wait period of 10 seconds allowed all processes to stabilise;

(5b) PSUTIL data capture commenced;

(5c) The SuperMediator command in the first terminal window was executed;

(5d) After 15 seconds (300 timer iterations) the capture was stopped;

(5e) The highest CPU utilisation over the 15 second period was recorded;

(6) The python script was executed a total of 10 times, to minimise the noise effect from background processes;

(7) The lowest six CPU loads were recorded;

(8) The test was repeated for all three templates.

SuperMediator templates can be found in Appendix C.

The python script can be found in Appendix D.

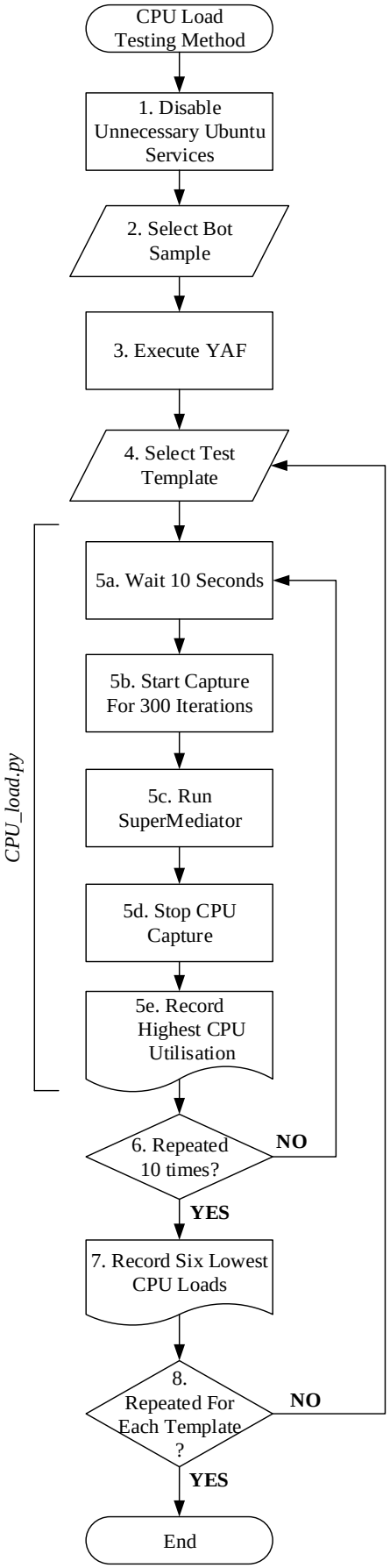


Figure 18. Flow diagram of the CPU load test.

The python script used the system processor attribute from the PSUTIL library to measure the impact of YAF upon CPU in the Ubuntu testing VM.

4.8.9 TEST #3 - CPU LOAD TESTING ANALYSIS

To quantify the differences in impact upon the underlying testing CPU from each template the mean CPU load was compared for each template. To minimise the noise effect from background Ubuntu processes upon CPU, each template was executed 10 times with the lowest four CPU loads rejected. The standard error of the highest six CPU loads allows experimental error values to be calculated.

As with the processing time test, it was found during the experiment design phase that background processes in the Ubuntu testing VM created background noise that impacted the times taken to run the processing tests. It was considered unfeasible to eliminate all background noise, hence the same noise reduction steps were taken as outlined in the processing time test, above.

4.9 BotProbe Performance Results

4.9.1 TEST #1 - PROCESSING TIMES TESTING RESULTS

10 bot sample datasets were tested against each template - BotProbe, extended BotProbe and simulated NetFlow v5. Table 22 compares the processing times for each dataset; (1) the BotProbe IPFIX with NetFlow v5 and (2) the extended BotProbe with NetFlow v5.

TABLE 22. A COMPARISON IN PROCESSING TIMES BETWEEN IPFIX AND NETFLOW V5

Dataset	BotProbe vs NetFlow v5	Extended vs NetFlow v5
CTU3	12.63% ± 0.22%	-25.51% ± 0.13%
CTU8-9	29.35% ± 0.27%	10.99% ± 0.76%
CTU110-4	29.42% ± 0.42%	11.86% ± 0.54%
CTU127-2	40.85% ± 0.01%	17.85% ± 0.05%
CTU141-2	32.96% ± 0.46%	2.50% ± 0.24%
CTU142-1	32.92% ± 0.17%	3.28% ± 0.10%
CTU144-1	31.92% ± 0.13%	13.29% ± 0.06%
CTU147-1	18.92% ± 0.21%	4.54% ± 0.08%
CTU148-1	28.42% ± 0.14%	2.39% ± 0.04%
CTU149-1	9.94% ± 0.61%	-15.50% ± 0.82%
MEAN	**26.73% ± 0.31%**	**2.57% ± 0.42%**

4.9.2 TEST #2 - DATA VOLUME TESTING RESULTS

The same 10 bot sample datasets were tested against each templates - BotProbe template, extended BotProbe template and simulated NetFlow v5. Table 23 compares the sizes of the total files output. The BotProbe template is also compared to the size of original PCAP for each dataset.

TABLE 23. A COMPARISON IN DATA VOLUMES BETWEEN IPFIX, NETFLOW V5 AND PCAP

Dataset	BotProbe vs NetFlow v5	Extended vs NetFlow v5	BotProbe vs PCAP
CTU3	11.54% ± 0.00%	-116.71% ± 0.00%	94.65% ± 0.00%
CTU8-9	11.94% ± 0.00%	-17.44% ± 0.00%	96.01% ± 0.00%
CTU110-4	17.08% ± 0.00%	-23.78% ± 0.00%	97.32% ± 0.00%
CTU127-2	13.32% ± 0.00%	-153.48% ± 0.00%	94.80% ± 0.00%
CTU141-2	13.54% ± 0.00%	-182.27% ± 0.00%	98.24% ± 0.00%
CTU142-1	18.35% ± 0.00%	-147.02% ± 0.00%	98.76% ± 0.00%
CTU144-1	12.17% ± 0.00%	-30.68% ± 0.00%	89.25% ± 0.00%
CTU147-1	10.47% ± 0.00%	-4.82% ± 0.00%	93.61% ± 0.00%
CTU148-1	13.18% ± 0.00%	-97.81% ± 0.00%	74.80% ± 0.00%
CTU149-1	19.04% ± 0.00%	-48.60% ± 0.00%	92.09% ± 0.00%
MEAN	**14.06% ± 0.01%**	**-82.26% ± 2.05%**	**92.95% ± 0.22%**

4.9.3 TEST #3 - CPU LOAD TESTING RESULTS

Four bot samples of various flow sizes (CTU148_1(Zusy), flows = 172287; CTU149_1(Kelihos), flows = 235287; CTU3_1(Kelihos), flows = 318602 and CTU145_1(uTorrent), flows = 411928) were processed with the maximum system CPU utilisation shown in Figure 19.

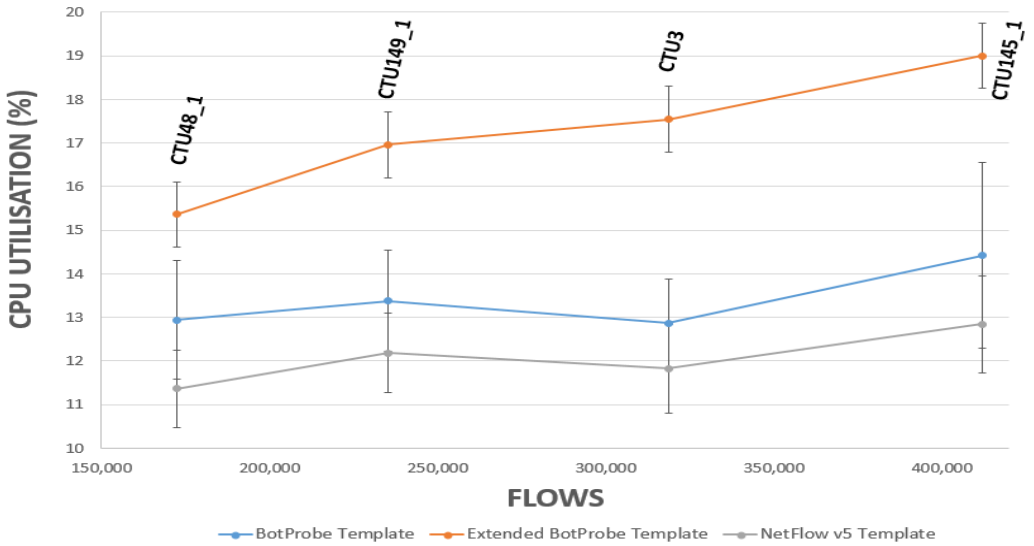


Figure 19. CPU utilisation for each of the three templates.

4.10 Discussion of BotProbe Performance

4.10.1 TEST #1 - PROCESSING TIMES

The BotProbe template demonstrated an average reduction in processing time of 26.73% ± 0.03% over NetFlow v5. This was higher than the anticipated reduction of 10%, from the BotProbe template being 43 bytes in size, compared to 48 bytes for NetFlow v5. Some of this additional reduction may result from the fields chosen to simulate the NetFlow v5 template, where some fields may be quicker to process than others. However, experimental verification of this was not undertaken because the method of comparing against a simulated template will only ever provide approximate results. The extended BotProbe template demonstrated an average reduction in processing time of 2.57% ± 0.04% over NetFlow v5. This was an unexpected result, as the extended BotProbe template contains more fields than the NetFlow v5 template, which implies it should take longer to process. Again, this may be due to the fields chosen to simulate NetFlow v5. The extended BotProbe template exhibited a range in processing times, from -25.51% ± 0.13% to 17.85% ± 0.05%. It was expected that this variance is due to the nature of the bot samples. When a sample contains a higher quantity of application data, the flow requires more processing, which increases processing times. The bot samples, chosen at random for testing, contained a range of application protocols in differing quantities. Hence, a range in processing times was not unexpected. Furthermore, if an EE was specified for export in a template but the protocol is not present in the flow, rather than exporting null data against this field, YAF does not export any data for this EE, thereby improving processing times. This suggests that any number of EEs can be included in the template without impacting processing time if they are not present in the flow. Therefore specific IPFIX templates could be created to target specific bot families, such as DNS bots.

4.10.2 TEST #2 - DATA VOLUMES

The BotProbe template demonstrated an average reduction in data volumes of 14.06% ± 0.01% over NetFlow v5. This was slightly higher than the 10% reduction expected, from the BotProbe template being 43 bytes in size compared to 48 bytes for NetFlow v5. However, this confirmed that the field sizes in the simulated NetFlow v5 template closely resembled the actual NetFlow v5 field sizes. The BotProbe template exhibited a range in reduction volumes from 10.47% ± 0.00% to 19.04% ± 0.00%. This was anticipated to be due to how YAF processes flow aggregation, and the differences in aggregation between the IEs in the BotProbe template against those in the simulated

NetFlow v5 template. The extended BotProbe template exhibited an increase in traffic compared to NetFlow v5, from 4.82% ± 0.00% to 182.27% ± 0.00%. This increase may be due to bot samples containing varying application protocol data, where EE string lengths will vary by both protocol and by bot sample context. However, as demonstrated in the processing times tests above, despite an increase in data volumes, the templates were quicker to process. This raises the possibility of further performance improvements by constructing EEs that export more specific variable length fields. Comparing the BotProbe template with the original raw PCAP capture files exhibited a reduction in data volumes ranging from 74.80% ± 0.00% to 98.76% ± 0.00%, depending on the context of the flows. *Hofstede et al.,* (2014) indicate an expected reduction from IPFIX in the order of 1/2000th of the original PCAP size. However, Hofstede does not specify IPFIX elements, so their results can only be taken as an overall order of reduction. The BotProbe template may not have achieved these reductions as it is tailored for a specific application. Even so, for a CSP capturing TBs of PCAP data, the order of volume reduction from the BotProbe template could result in GB quantities of data. With European communications providers required to retain connection data for between 6 to 24 months (*EC Data Retention Directive 2006/24/EC, 2006*) this is a saving in storage requirements.

4.10.3 TEST #3 - CPU LOADS

There were no measureable impact upon CPU load between the BotProbe template and NetFlow v5. The BotProbe template averaged at 13.4% ± 0.7% impact, whilst the NetFlow v5 averaged at 12.1% ± 0.6% impact, which was within experimental error margins. The extended BotProbe template averaged at 17.2% ± 0.8%; a marked increase in CPU load, possibly due to the additional software plugins required when capturing EEs. In comparison, the CPU load benchmark with no IPFIX export was 5% ± 0.0%. These results were obtained in a Ubuntu 14.04 LTS desktop VM, with four 2.6GHz processors with 2.9GB RAM. Scaling this up to running YAF on a high end server should expect minimal CPU impact. These minimal CPU impact results suggest the feasibility of running IPFIX export at 1:1 capture rates on low specification, low powered devices, such as those found in the IoTs. In high-speed data networks, NetFlow is typically configured to sample at rates between 1:500 and 1:2000; not only compensating for a lack of flow aggregation which produces high volumes of capture data, but is also to conserve device power. Sampling is detrimental to anomaly detection (*Mai, et al., 2006*) as it is more likely to miss small, infrequent bot keep-alive packets.

4.11 Summary

Cisco created NetFlow as a protocol to support network management. NetFlow v5 was the traffic capture mechanism for many botnet detection algorithms, despite drawbacks being known when applying the protocol to security threat detection (*Santos, 2016; Trammell and Boschi, 2011; Gates et al., 2004*). A primary weakness is the rigidity of the 18 field NetFlow v5 template, of which eight fields offer little value in botnet detection. NetFlow v9 has its own drawbacks. The protocol is proprietary, limited to 79 IEs and lacks support for EEs, restricting it to capturing only packet header information with no functionality to extract payload data (*Patterson, 2012*). This has forced many researchers to supplement traffic capture requirements with PCAP (*Sperotto, et al., 2010*).

The hypothesis of this research predicts that IPFIX offers advantages over NetFlow in botnet detection. Chapter 3 outlined tangible benefits of IPFIX, such as security by design and scalability through support for complex protocols such as IPv6 and MPLS, whilst also being a ratified international standard. The contribution from this chapter comes from empirical evidence that template extensibility, variable length fields and enterprise elements all hold benefits over both NetFlow v5 and PCAP, addressing research objective #2 to create an IPFIX template for botnet traffic capture.

The challenge for any traffic capture mechanism is to move away from capturing big data volumes, towards capturing a smaller yet manageable dataset to ease data analysis. The BotProbe template demonstrated an improvement over NetFlow v5 of an average of a 26.73% ± 0.03% reduction in processing time and an average of 14.06% ± 0.01% reduction in data volumes. The BotProbe template demonstrated as much as a 98.76% ± 0.00% reduction in traffic volumes over PCAP. The extended BotProbe template expanded the capture mechanism to include application protocol information, providing contextual botnet information upon the network layer information from the packet header. With IPFIX as a capture mechanism, previous botnet detection experiments should expect an improvement in both capture speed and reduction in data volumes when run against the original capture method, without the need to amend the variables used by the detection algorithms. A reduction in data volumes translates as a benefit to communication providers bound by European law to retain connection information for a period of time.

The evidence provided in this chapter suggests that IPFIX has the potential to advance botnet detection. The limitations of other traffic capture mechanisms have confined the variables used in detection algorithms to the few traffic characteristics

that the protocols can capture. IPFIX turns this around, allowing the algorithms to dictate the variables that are required for capture, rather than the capture mechanism determining the available variables for the algorithms. This should permit the creation of new botnet detection algorithms based on new traffic attributes.

Having provided empirical evidence to demonstrate that IPFIX offers advantages over both NetFlow and PCAP, the next chapter uses these IPFIX templates as a foundation upon which to construct a framework to incorporate IPFIX export into a CSP environment.

5 BotStack: A Novel IPFIX Framework

5.1 Introduction

Building upon the evidence provided from the previous chapter that IPFIX provided reductions in both data volumes and processing times over NetFlow v5, this chapter describes an architectural framework for incorporating an IPFIX capture element into the cloud stack. Whilst previous academic studies have suggested various cloud stack frameworks, many of which are covered in this chapter, no literature could be found incorporating IPFIX within the cloud stack.

BotStack, the framework proposed within this chapter forms the foundation of a proof of concept demonstrator platform for future work in optimising botnet traffic capture and detection methods. The platform is constructed from open source tools that are common to CSP environments; thereby easing the migration process from existing cloud environments to this novel IPFIX platform. Open source tools are selected as the building blocks for BotStack because not only are they commonly found in CSP environments, but an open source approach permits modification to software to overcome any interoperability issues, or enhancements.

5.2 Design Considerations for IPFIX Export in a CSP Environment

A cloud is a multi-tenanted environment with high privacy expectations. The design requirements of a cloud provider infrastructure provide additional challenges over a traditional network. *Lenk, et al.,* (2009) describe three basic components of a cloud infrastructure: (1) the physical resource, (2) the virtual resource and (3) the management front-end API which allows automate setup and tear-down of virtual machines, failover, demand scalability and OS provisioning. Effectively, a cloud environment can be described as a virtualised guest-layer running on top of a physical network-layer.

In this research project, the CSP framework fundamentally follows an Infrastructure as a Service (IaaS) model, in which guest operating systems share an underlying infrastructure, but are partitioned from other tenant VMs. NIST, the National Institution of Standard and Technology, defines IaaS as where "the capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications." (*Mell and Grance, 2011*).

Within these design constraints the following assumptions were made:

- The framework provisions the design principles of co-residency, where tenant isolation is provided upon a shared underlying infrastructure; and data integrity, where data privacy and integrity are provided through minimal interaction with tenant data traversing the CSP infrastructure;
- The framework facilitates support for leading-edge technology, where the provider recognises cost reductions and operational enhancements through the inherit benefits from superior technologies such as IPv6 (*Díaz, Martín and Rubio, 2016)*;
- The framework is to be constructed from open source technology wherever possible, thereby allowing software modification between existing and future framework elements.

Multiple component elements are required to facilitate IPFIX export within a CSP infrastructure. Figure 5 identifies four elements that are necessary in flow monitoring; the probe, the collector, data storage and data analysis. The following section provides justification for each framework element selected for traffic data capture. This includes not only IPFIX export and collection, but consideration is given to the elements that are necessary in the provision of a virtual infrastructure across which IPFIX export is supported. Less consideration is given to the physical infrastructure and data storage, as this modular framework should be capable to being overlaid on top of the CSPs existing infrastructure, whilst data analysis is considered in more detail within future work in chapter 7.

5.2.1 HYPERVISOR

Machine virtualisation allows a single physical machine to host multiple, heterogeneous Operating Systems upon the same hardware (*Garcia-Valls, Cucinotta and Lu, 2014*). In an IaaS model it is not uncommon for VMs from several various tenants to be co-located on the same physical hardware, where each VM requires isolation from its neighbours. Virtualisation is provided by a Virtual Machine Monitor (VMM), also known as a hypervisor. The hypervisor is a software component that isolates multiple VMs whilst managing the sharing of physical host's resources, thereby allowing each emulated VM to act as a stand-alone, tightly isolated container. The vulnerabilities described in chapter 2 typically occur by exploiting weaknesses in hypervisor software code. Hypervisors can be broadly split into two categories; type-1 bare-metal hypervisors that sit directly on the host hardware, and type-2 hosted hypervisors that sit as applications in the host OS. Examples of type-1 hypervisors include Xen, VMware ESXi and Microsoft's Hyper-V, whilst type-2 hypervisors include Linux KVM (Kernal-based Virtual Machines), VMware Player and Oracle's VirtualBox. Figure 20 compares the benefits and disadvantages of four of the market leading hypervisors. As type-1 hypervisors typically have a higher performance than type-2 hypervisors, KVM, as a type-2 hypervisor, is not considered for the framework. Of the remaining three hypervisors, Xen is open source, available under a GPL (General Public License).

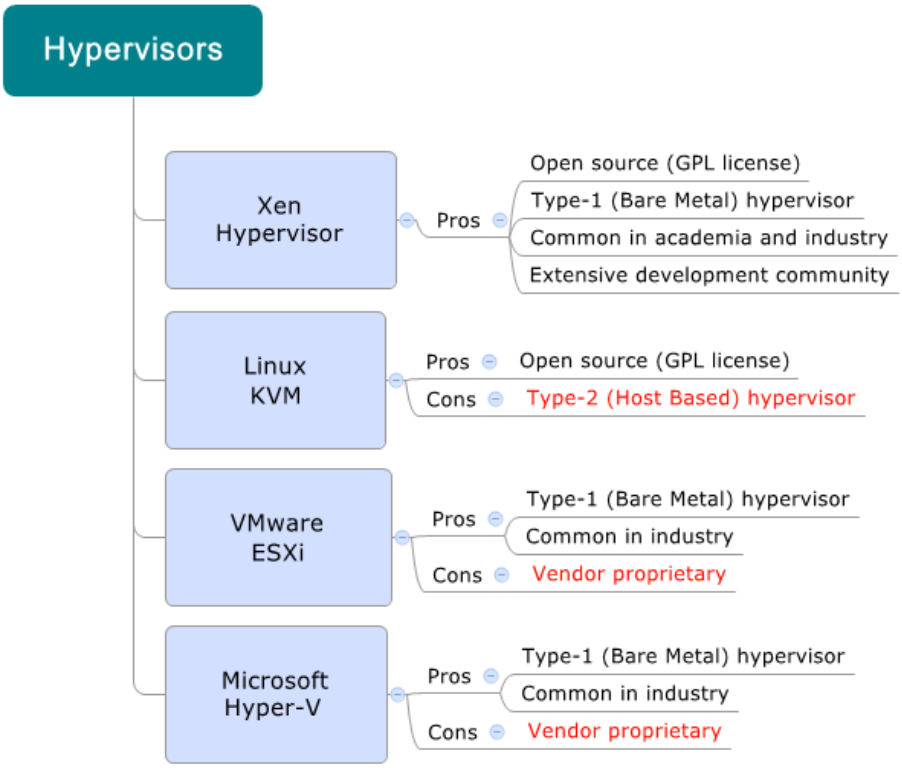


Figure 20. A comparison of hypervisors.

Xen hypervisor is available as either a standalone hypervisor (*Linux Foundation, 2013*), or within the XenServer project from Citrix (*Citrix Systems Inc., 2015*). The XenServer project is an open source project that provides an out-of-the-box solution bundle, including Xen hypervisor, Open vSwitch, Xen Management software and a pre-installed CentOS operating system. This includes all the major elements for an IPFIX framework. Initially the BotStack framework was designed with XenServer v6.2.0 as its foundation, but during the build of the infrastructure several issues were encountered. More detailed descriptions of the issues encountered were presented at BotConf 2015 (*Graham, Winckles and Sanchez, 2015b*). In summary, XenServer v6.2.0 ships with an older version of Open vSwitch (OVS) which does not support IPFIX. Upgrading OVS requires an upgrading the XenServer CentOS operating system to 64-bit. In order to keep the OS footprint as small as possible, the incumbent CentOS 5.6 in XenServer has a reduced feature set which prevented the upgrade of the OS. Citrix subsequently released XenServer v6.4.94, known as XenServer Creedence, [*sic*] which includes CentOS v5.10 and a version of OVS that supports IPFIX. Unfortunately in testing, OVS failed to provide any IPFIX timestamps. Also, OVS flow aggregation did not work, which meant huge numbers of individual data flows instead of a reduced number of aggregated flows. Due to these issues, XenServer was rejected, necessitating a custom build framework using the standalone Xen hypervisor, on top of a Ubuntu OS.

As a type-1 hypervisor, Xen runs directly on the hardware, booting directly from BIOS. With Xen, the privilege Domain-0 (Dom0) creates and destroys the emulated VMs that run in the abstracted guest Domain-U (DomU), and controls the DomU access to the underlying hypervisor and physical resources. Xen is a popular choice in industry and is prominent amongst CSP. Key members of the Linux Foundation Xen Project[7] include Amazon AWS and Citrix. Xen is also more prominent in academic research, compared with other open source hypervisors such as KVM. Xen was the hypervisor of choice in IaaS platform research, including OpenEdge (*Kunz, et al., 2016*); Apache CloudStack (*Kumar, et al., 2014*); OpenStack (*Sefraoui, Aissaoui and Eleuldj, 2012*); Nimbus (*Keahey, 2009*); Eucalyptus (*Nurmi, et al., 2009*) and Cumulus (*Wang, et al., 2008*). When *Jasti, et al*, (2010) analysed security in multi-tenanted cloud environments, they built their test environment on top of a Xen hypervisor. *Garcia-Valls, Cucinotta and Lu* (2014) describe two operational modes of Xen hypervisor. Para-virtualisation mode is a higher performance mode as no hardware platform emulation is required, making it more suitable for real-time cloud

[7] https://www.xenproject.org/project-members.html

computing since it identifies the specific components of an operating system that have to be virtualised in order to optimise performance. Para-virtualisation has challenges such as virtualizing and sharing memory between guest operating systems. Also, a guest OS may require modification to make it aware that it is running in a virtualised environment, so that it can maintain direct communication with the hypervisor via hypercalls. Full-virtualisation mode, or Hardware-assisted Virtual Machines (HVM), allows full emulation of hardware attached devices, such as network adaptors, without the need to modify the guest OS. As each VM has its own virtual BIOS, each VM is un-aware it is being emulated on top of a host device. This makes it possible to run multiple operating systems, even heterogeneous ones, on the same hardware. Whilst HVM more closely resembles the complete isolation of a physical server, it is both slower and more expensive. CSPs, such as Amazon AWS, tend to run the faster para-virtualisation, particularly when running Linux. However, hypercalls direct to the hypervisor allow malware to exploit the type of vulnerabilities outlined in chapter 2.

The Xen hypervisor was selected upon which to base BotStack, so as to maintain an open source infrastructure.

5.2.2 HYPERVISOR MANAGEMENT

A CSP typically uses a management API toolstack to manage hypervisor events, such as the setup, monitoring and tear-down of individual VMs, or the provisioning of an OS (*Lenk, et al., 2009*). There are two options available for Xen management through APIs (see Figure 21). By default, Xen ships with the XL toolstack; a lightweight minimal toolstack based on the xenlight library (libxl). XL has a limited command set and has been designed for managing single host environments, replacing the now deprecated XEND toolstack.

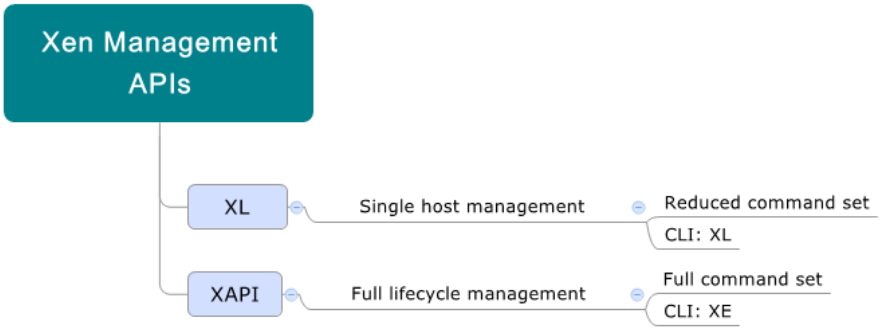


Figure 21. A comparison of Xen toolstacks.

Citrix recommends XAPI (Xen Application Programming Interface) for managing virtual device pools. XAPI was originally created by Citrix specifically for use with XenServer. XAPI is now open source and supports the full life cycle management that a CSP might be expected to perform, such as managing storage repositories, VM states, device pools and high availability.

XAPI was selected for BotStack to manage the Xen hypervisor environment as it provides a higher functionality toolstack than XL.

5.2.3 VIRTUAL MACHINE MANAGEMENT

The XAPI toolstack is a command line interface. Management of CSP virtual environments can be complex, so a GUI is preferred. The Xen Project lists several GUI packages that integrate into the toolstack API, as outlined in Figure 22. Xen Orchestra was not able to establish a connection to the test bed. The Xen Project suggests two other GUIs that were not considered for the framework; ConVirture was chargeable after a 30 day trial period and Zentific was no longer available for download. Both XenCentre and OpenXenManager, an open source copy of XenCentre, provide identical functionality.

XenCentre was selected for BotStack, as OpenXenManager was found to occasionally freeze, requiring a complete system reboot.

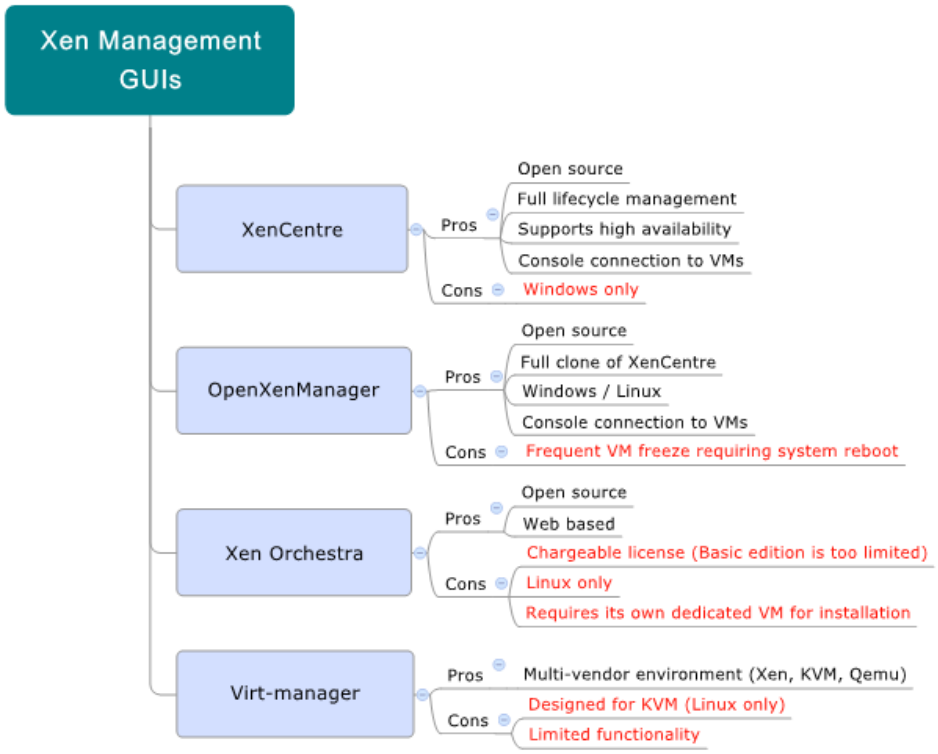


Figure 22. A comparison of management GUIs.

5.2.4 VIRTUAL SWITCH

Advantages to cloud providers of virtual networking include energy efficiency savings (*Beloglazov and Buyya, 2010*), dynamic allocation of resource and bandwidth (*Guo, et al., 2010*) and optimisation of VM placement and traffic flow (*Fang, et al., 2013*). With type-1 hypervisors, such as Xen hypervisor, virtual machines share the resources of their host server. In a physical network, hardware devices typically use network interface cards to connect to hardware switches. In a virtual network, virtual devices can be connected, via a virtual network interface to software switches, known as virtual switches (vswitches). The performance of vswitch CPUs now match the performance of physical switch CPUs (*Pettit, et al., 2010*) whilst having the advantages of being virtual. Of the four data centre vswitches outlined in Figure 23, only Open vSwitch (OVS) is open source, making OVS popular within academic research. By default, OVS operates in bridging mode using the built-in bridging functionality of the host Linux distribution. Alternatively, OVS can be configured to operate as a fully functional switch in switching mode. In either mode, there is a negligible difference in network throughput (*Pettit, et al., 2015*) or CPU impact (*Rintalan, 2011*), however switching mode supports VLANs, quality of service, access control lists and port bonding which may occur within a CSP infrastructure.

OVS was selected for BotStack as it integrates with all major hypervisors, and provides support for a range of monitoring protocols, including NetFlow v5, NetFlow v9 and IPFIX (*Pfaff, et al., 2015*).

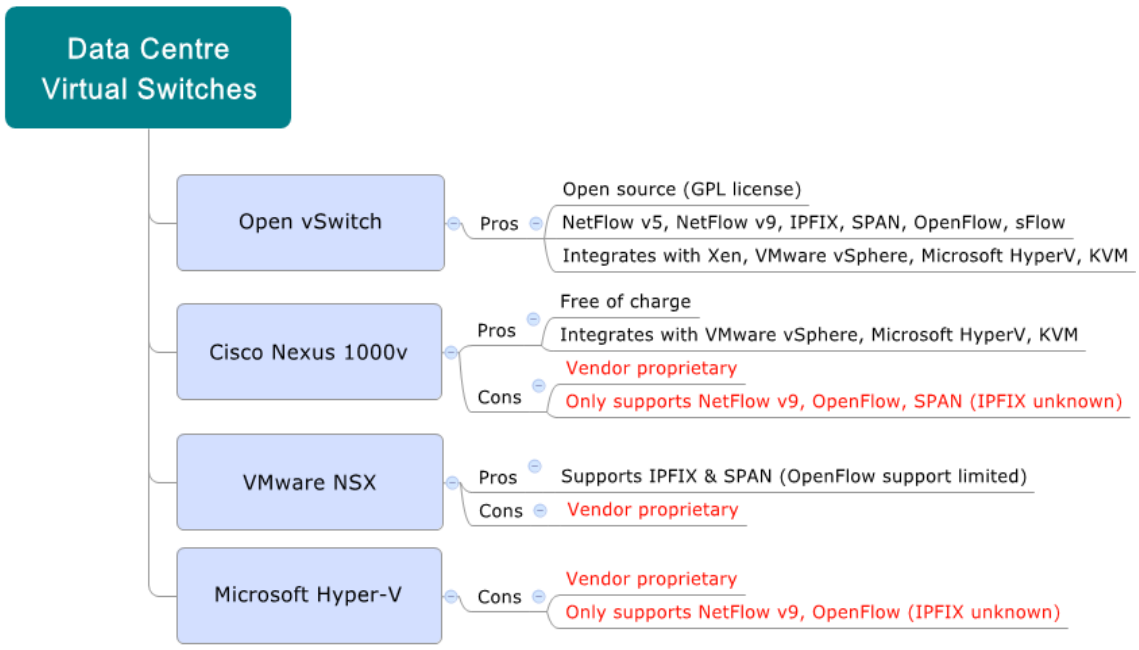


Figure 23. A comparison of data centre virtual switches.

Additionally, OVS has its own flow management controller which uses OpenFlow. This would allow a CSP to interface into a Software Defined Networking architecture, for dynamic on-the-fly network configuration to contain, or divert, a botnet. OpenFlow can be disabled should a CSP favour an alternative third party controller.

5.2.5 FLOW EXPORTATION

Two of the essential components in flow export are the exporter probe and collector (Figure 5). *Hofstede, et al.,* (2014) lists a number of commercial and open source flow exporters, each with varying degrees of support for IPFIX features. In order to take advantage of the IPFIX benefits outlined in chapter 3, an IPFIX exporter should support features that include: (1) IPFIX flow aggregation to reduce the overall volume of exported data; (2) IPFIX template customisation with support for both IEs and EEs; (3) the ability to periodically dump captured flows to a data file for analysis; and (4) standards-based IPFIX security features including flow integrity and obfuscation.

OVS claims to export NetFlow v5, NetFlow v9 and IPFIX. OVS is configurable to capture packets on a physical interface (PIF) via an network interface card, and virtual interfaces (VIF) via a network tap. These features make OVS suitable for deployment within the framework. During the build of the infrastructure doubts were raised over the OVS capability to support for IPFIX export. Several IPFIX collectors, including YAF, nProbe, Plixer's Scrutinizer, IPFIXcol and nfdump, would not recognise the IPFIX stream format exported from OVS. In both OVS bridging mode and switching mode, IPFIX was found to be missing timestamps and appeared not to apply flow aggregation, resulting in an overload of traffic at the collectors. This was reported to the OVS developers, who did not perceive this to be a priority fix. Furthermore, as OVS did not support template customisation, OVS was retained in BotStack as the virtual switch, but was not used to perform IPFIX export functionality.

Of the open source IPFIX exporters in Figure 24, only two probes support both IE and EE template customisation: nProbe (*Deri, 2003*) and YAF (*Inacio and Trammell, 2010*). The process for template configuration is equally straightforward in both YAF and nProbe, where template fields for capture can be specified in the command line syntax at exectution. YAF has been designed to specifically conform to IPFIX compliance, making YAF the only exporter that supports RFC-6313 for structured data, an advantage when querying data. The cost of IPFIX compliance means that, unlike nProbe, YAF does not support NetFlow. NProbe has an advantage in that it is both an exporter and collector in one. NProbe supports a greater number of IEs than

YAF, although nProbe's IEs are predominantly flow contextual rather than traffic contextual data, as outlined in Table 3. Both exporters support EE export through software plug-ins, although information on how to create new EEs was lacking in both cases. nProbe is available at no cost through an academic license, although full functionality required a commercial license.

Argus is a well-known network audit and traffic analyser that supports its own version of NetFlow. Argus documentation states that it only supports a subset of IPFIX, so was not considered in this research. Bro is also a well-known tool for IDS, which supports IPFIX. However, Bro is primarily a SEIM solution that monitors security event and policy violations, but is not capability of providing the level of data correlation required by this research project.

A review of IPFIX literature revealed that YAF and nProbe are equally the most commonly used exporters in academic research. A trend was observed for nProbe being the primarily exporter between 2004 and 2008, with YAF overtaking as the preferred exporter after 2009. When *Velan, Jirsik, Čeleda* (2013) tested their HTTP header parsing algorithms, the only exporters that supported HTTP IEs were nProbe and YAF. The algorithm they designed for nProbe was marginally quicker than their regex algorithm tested with YAF, however they failed to provide a head to head comparison when no algorithms were present, providing no reason to favour one exporter over the other. No reliable evidence could be found comparing the performance of nProbe with YAF.

When *Brockhus* (2015) compared a number of flow exporters, YAF exported 1229 flows against nProbe's 924 flows over the same traffic capture. They do not explain this difference in results. Furthermore, the reliability of this data is questionable as the research has yet to be published in a reputable journal. The performance speed of YAF against nProbe was not tested during this research, as the IPFIX framework requirement is for a full-featured IPFIX exporter, rather than the fastest. When *Rincón, et al.,* (2015) used IPFIX to capture TCP connections they chose YAF because it provides a MySQL mediator to their analysis software.

Whilst either nprobe or YAF were fit for purpose as the IPFIX export element of BotStack, YAF was selected due to its support for a larger number of EEs by default.

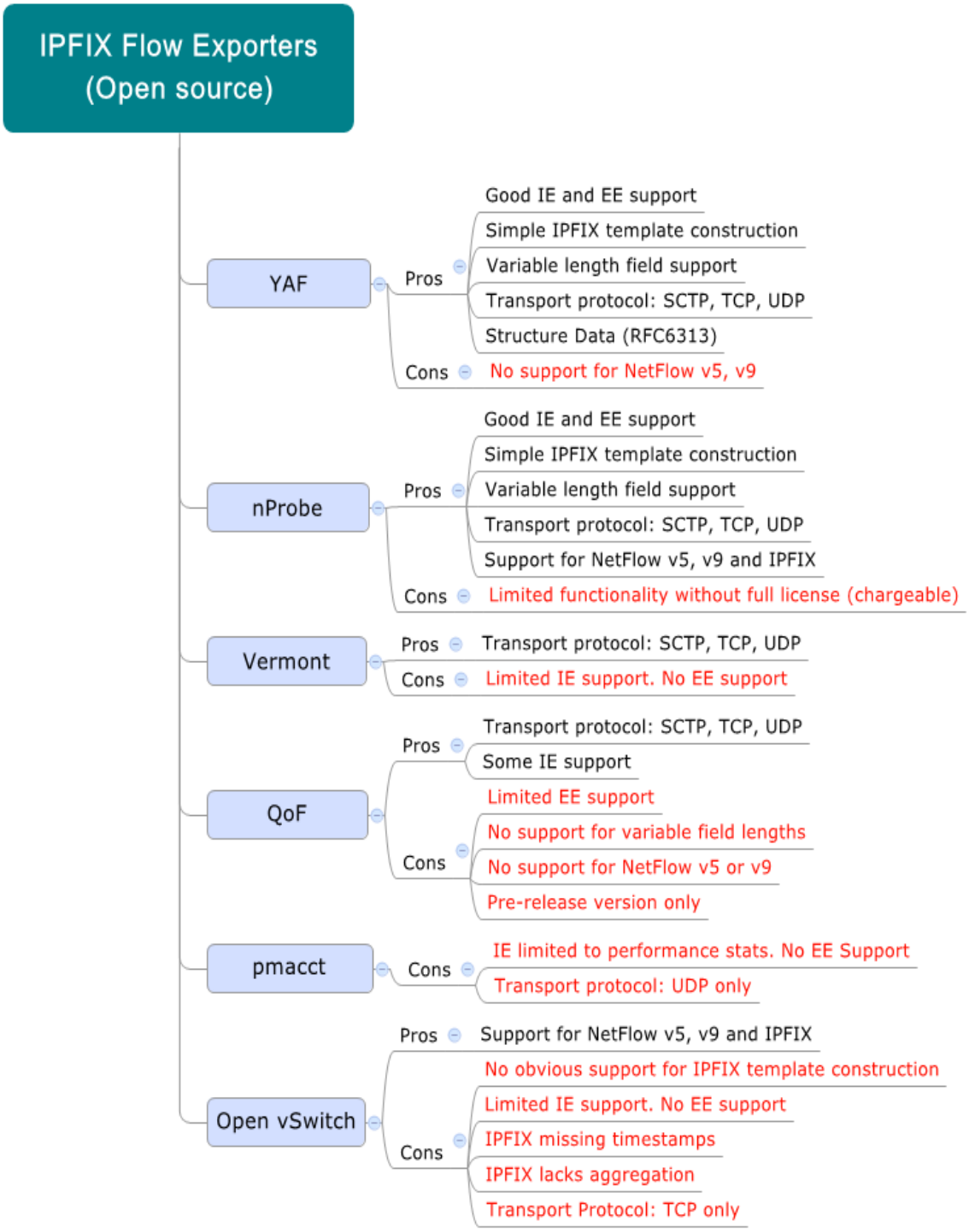


Figure 24. A comparison of open source IPFIX flow exporters.

5.2.6 FLOW COLLECTION

The YAF IPFIX exporter is a probe that sits in a traffic stream capturing data. It then encapsulates any captured data into the IPFIX protocol for transport. This IPFIX stream is then sent to a collector, which converts these exported flows into formats required for storage and analysis. The selection of a flow collector is often based upon the desired storage format (*Hofstede, et al., 2014*). Again, *Hofstede, et al.*, (2014) lists commercial and open source flow collectors. Open source collectors are compared in Figure 25. IPFIX collectors with noteworthy support for IE and EEs include nProbe, IPFIXcol and the SiLK suite. When *Velan*, (2013) compared IPFIX collectors, nProbe was ignored as it is an exporter by design, rather than a collector. Velan went on to compare nfdump, SiLK and IPFIXcol; concluding that IPFIXcol was the quickest and most flexible. However, this is subject to bias as IPFIXcol was created by Velan as a PhD project. IPFIXcol was found to be overly complicated with little or no documentation, restricting its application to the framework.

YAF is part of the Network Situational Awareness (NetSA) security tool suite developed by software engineers at CERT. By design, YAF exports IPFIX in a proprietary `.yaf` format, as this allows YAF to be feature rich, whilst letting a mediator convert the data to a human readable format. NetSA created SiLK as the collector for the YAF proprietary format. The drawback of SiLK is that it was designed as a network management analysis tool, so focuses heavily on flow statistics, rather than traffic content analysis. Another disadvantage is that SiLK stores data in a "packed" flat binary file format, which requires an additional NetSA tool called *rwcut* to analyse data. Rwcut is designed for high-level flow context analysis and does not have the functionality to perform detailed statistical tests such as frequency and correlation. Other NetSA created mediators that support `.yaf`, include yafscii and ipfixDUMP, however neither support outputting data as `.xls` or `.csv` format. The role of an IPFIX mediator is to provide federation of IPFIX messages, allowing anonymisation, filtering, translation and aggregation of IPFIX streams to one or more collectors (*Santos, 2016*). SuperMediator is an IPFIX mediator, created by NetSA, which imports `.yaf`, and outputs it to `.csv`.

SuperMediator was selected for BotStack to partner YAF as the IPFIX collector. Mediators are not designed for complex flow analysis. SuperMediator can be extended using python plugins, allowing support for additional IEs/EEs to be added. In this research, data analysis is performed manually, outside of the IPFIX framework requirements.

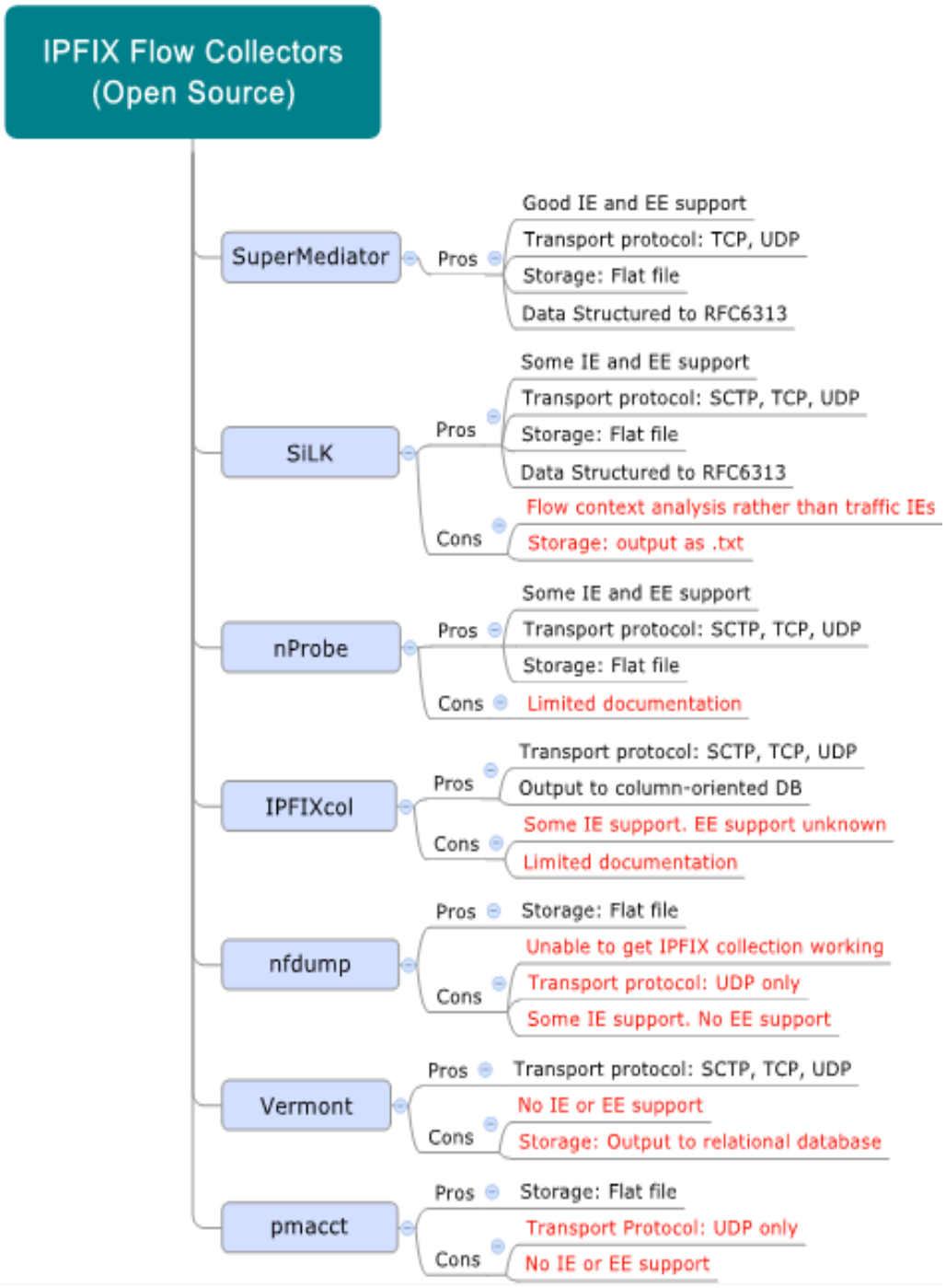


Figure 25. A comparison of open source IPFIX flow collectors.

5.3 Presenting BotStack: An IPFIX Framework for CSPs

The individual component elements identified in the above analysis for selection in the IPFIX framework for CSPs are presented in Table 24, with the logical representation of the BotStack framework architecture in Figure 26.

TABLE 24. BOTSTACK FRAMEWORK COMPONENTS

BotStack: An IPFIX Framework	
Host OS	Ubuntu 14.04 LTS
Hypervisor	Xen 4.4.0 (64 bit)
Hypervisor Management	XAPI Toolstack
VM Management	XenCentre v6.5
Virtual Switch	Open vSwitch v2.0.2
Flow Exporter	YAF v2.8.4
Flow Collector	SuperMediator v1.3.0

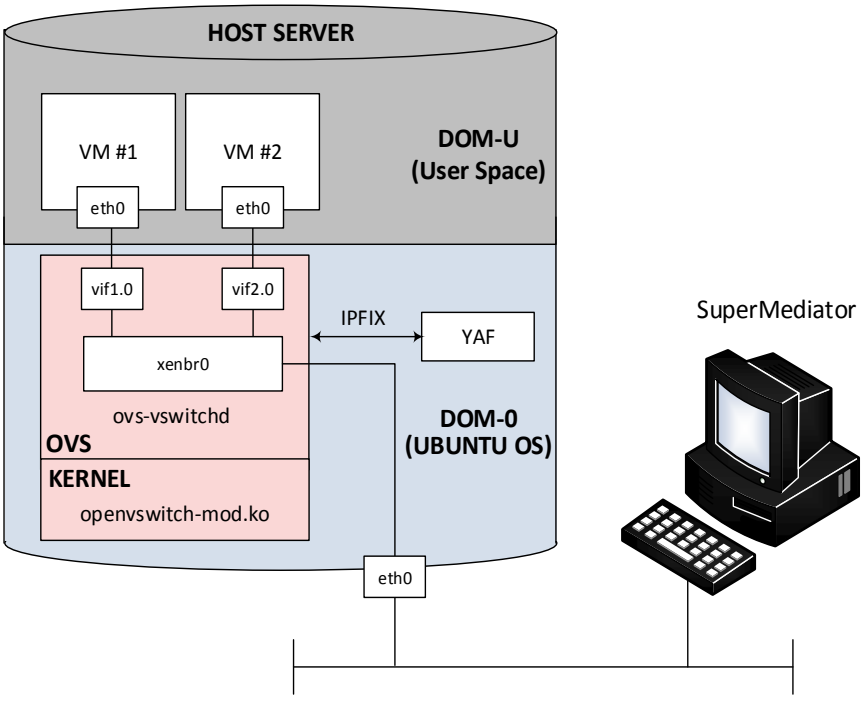


Figure 26. The logical architecture of BotStack.

5.4 Probe Positioning Test Methodology

A *"*network as a sensor*"* takes advantage of the distributed nature of networks to turn the networks themselves into proactive cyber event notification tools (*Cisco, 2015*). As a traffic capture mechanisms is just as vulnerable as any other networked device, due consideration to the siting of IPFIX export probes is necessary. RFC-7011 mandates that IPFIX transmissions include features for confidentiality, authentication and integrity, thereby mitigating the risk to IPFIX data of stream manipulation or interception (*Internet Engineering Task Force, 2013a*). Chapter 3 describes flow security features in more detail. Whilst these security features secure the data streams between devices, they do little to protect the probes themselves from attack. A probe sited within the unprivileged guest domain not only has the potential to imply surveillance rather than protection, but also presents a vulnerable attack surface to malware. Whilst a probe placed outside of the guest domain is less visible to an adversary, the risk profile of the probe increases should the adversary realise that the mechanism is specifically monitoring for malware. RFC-7011 (*Internet Engineering Task Force, 2013a*) makes recommendations to mitigate DDoS attacks on probes, however, a probe may still be vulnerable to timer misalignment attacks should malware be able to force clock changes upon the host device.

Catchment area is another consideration in probe placement. Flow is a *push* technology, exporting only data that passes through a probe. If a probe is not within a data stream it will not export this data. A single sampling point may lack visibility of key data streams, or create a congestion point in the network. Therefore multiple sampling points are required across the entire infrastructure where botnet activity is being monitored. Multiple probes not only present malware with a larger attack target surface, but increase the complexity of data coordination during analysis. A balance must be struck between the number of probes and their catchment area. *Hofstede, et al.,* (2014) state that the deployment of packet capture probes in virtual networks is similar to deployment in wired networks. They describe two modes in which probes can be positioned. Firstly, inline - where the probe is directly connected to the stream by a passive network tap that captures traffic at line speed without introducing delay. Secondly, mirrored - where packet forwarding devices, such as switches, mirror packets from one port for capture on another port. This means that a probe with a layer 2 vantage point on a switch, only captures traffic from mirrored (or SPAN) ports. Whereas a probe with a layer 3 vantage point can monitor blocks of IP addresses, thereby capturing traffic from all ports (*Collins, 2014*). *Minarik, Vykopal and Krmicek* (2009) compared the efficiency of mirroring, such as used in NetFlow,

with direct tap connections, such as used by IPFIX. Connecting a NetFlow probe via a SPAN port dramatically decreased the data quality compared to the network tap connection. The tap connection captured more flows as it was able to capture both incoming and outgoing traffic on the segment, whilst the SPAN port was limited in which segment it could capture data from. SPAN also saw more packet duplication.

5.4.1 DATASET

The resultant dataset from this test was generated by systematically polling probes that were strategically distributed across the CSP infrastructure (Figures 30 - 34).

5.4.2 EQUIPMENT

A test network was constructed using the BotStack components outlined in Table 24. Two Dell PowerEdge R710 servers, each with four Intel Xenon 5160 3.0GHz CPUs and 8GB RAM, ran Ubuntu 14.04 LTS desktop operating systems. Xen 4.4.0, the XAPI toolstack, Open vSwitch v2.0.2 and YAF v2.8.4 were installed into Dom0. Both servers had three Ubuntu 14.04 LTS desktop VMs. Open vSwitch was configured to use network taps to capture traffic across all virtual ports and forward this to a capture probe. The network was configured as a flat 192.168.0.0/24 network. Figure 27 outlines vantage points for probe placement; where probes #1 and #2 are within tenant virtualised environments, #3 is located on the CSP LAN, #4 and #5 are on host servers, with #6 and #7 are on networked devices. Five different tests measured the visibility of each probe placement.

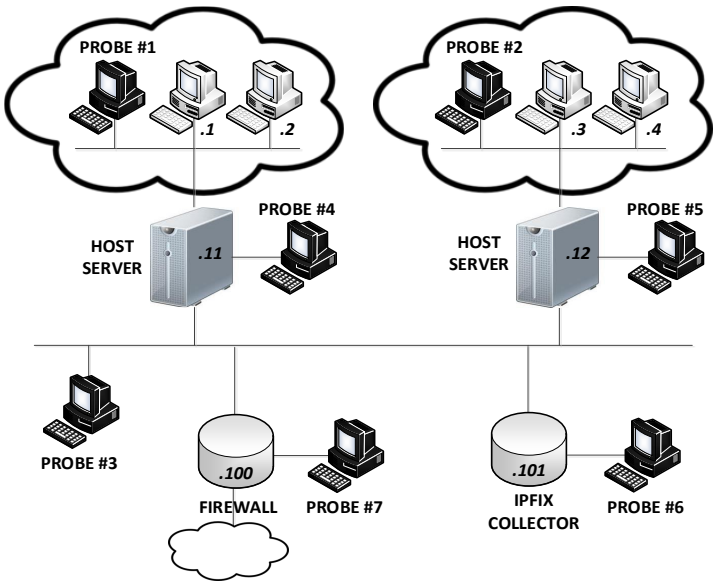


Figure 27. An illustration of potential probe vantage placements.

5.4.3 METHOD

The aim of this test was to empirically determine the optimum siting configuration of the IPFIX probes, to allow botnet propagation to be tracked across the CSP infrastructure with maximum network visibility for the least number of probe installations. The independent variables were the ICMP ping tests between devices. The dependent variables were the various available probe locations. Flow diagram Figure 28 summarises the systematic test method. The detailed method was:

(1) A YAF probe was installed into devices sited at the test vantage points;

(2) YAF was configured to export ALL network traffic as IPFIX:

```
# yaf --live pcap --in xenbr0 --out probeflow.yaf
```

(3) Each device was systematically pinged, so the probe captured resultant echoes:

```
# ping 192.168.0.x
```

(4) YAF was stopped;

(5) The YAF output file (`.yaf`) was saved for analysis.

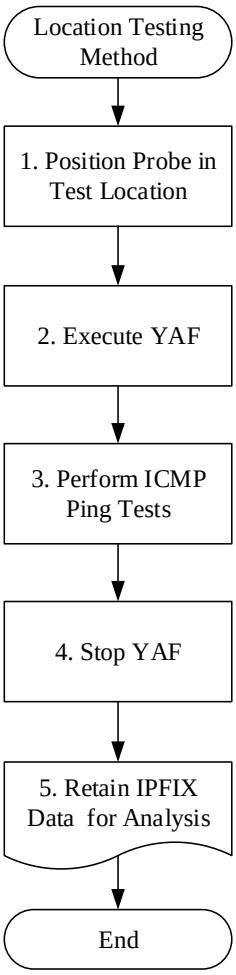


Figure 28. Flow diagram of the probe location optimisation test.

5.4.4 ANALYSIS

In order to empirically determine the optimum locations of the IPFIX probes, each probe placement point was analysed for its capability to capture systematic ICMP pings across an IP address range. To faciliate empirical calcuation, each device was assigned a costing weight as illustrated in Table 25, according to the importance of the network infrastructure the probe has visibility of.

TABLE 25. NETWORK LINK COSTINGS BY IMPORTANCE OF LINK

Network Link	Cost
VE to VE	50
VE to Server	10
Server to Server	5
Device to Device	1
VM to VM	0

These scoring metrics are graphically represented in Figure 29. The highest weighting was allocated to intra-VM connections, representing a malcious attack from one tenant upon another tenant. The next highest weighting was allocated between the tenant VE and the host server indicating either a VM escape or host escape. Monitoring server to server comminucations indicates a bot's potential to attack a neighbouring tenant. Monitoring device to device communication allows bot propagation to be tracked. As the aim was to track botnet propagation across a CSP network, no emphasis was given to VM-VM communication within a tenant's VE.

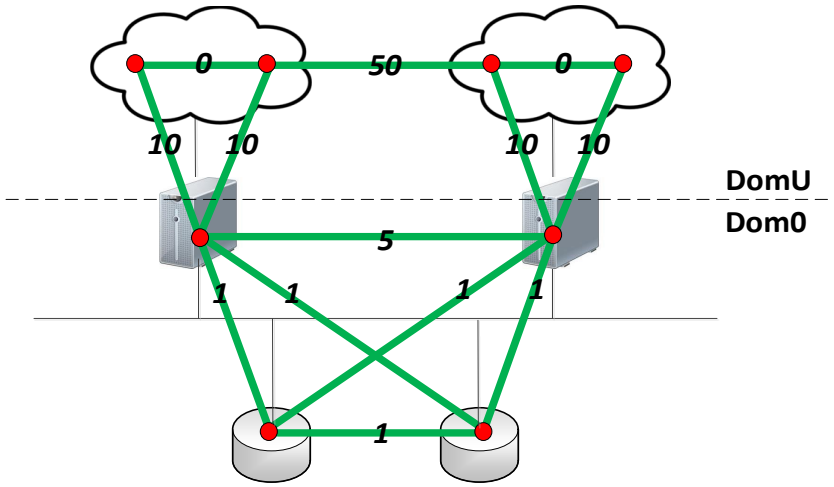


Figure 29. Network vista weightings, by importance in botnet communication detection.

5.5 Probe Positioning Results

Figures 30 - 34, below, display the observed device communication for each positioning test. Five individual positioning tests were conducted; with probes located at the potential vantage placements identified in Figure 27, in order to measure the network visibility across the network connection links outlined in Figure 29. Figures 30 - 34 each indicate the location of the capture probe(s) and network visibility is represented by red or green network node links; where a green link shows a successful ping that was detected by the probe, and a red links shows a successful ping that was not detected by the probe. Table 26 summarises the obtained weighted values for each test.

TABLE 26. WEIGHTED VALUES FOR ICMP PINGS, FOR EACH PROBE PLACEMENT TEST

	VE to VE	***VE to Server***	***Server to Server***	***Server to Device***	***Device to Device***	**TOTAL**
Test #1	0	0	0	0	0	**0**
Test #2	0	0	0	0	0	**0**
Test #3	50	2 x 10	5	2 x 1	0	**77**
Test #4	50	4 x 10	5	4 x 1	0	**99**
Test #5	50	4 x 10	5	4 x 1	1 x 1	**100**

Table 26 indicates that certain probe locations are sub-optimal for botnet communication traffic detection. Test #1 (Figure 30) was the benchmark test to understand visibility of traffic for probes located in the tenant virtual environment. Siting a probe in the tenant environment raises concerns around tenant privacy and surveillance; what other data is the probe capturing about the tenant environment besides botnet mitigation information. Furthermore, a probe in the tenant enviroment makes the probe highly visible. A malicious tenant may chose to disable the probe to turn the VM into an attack plane. Besides these drawbacks, Test #1 indicated that the probe had no visibility of communication traffic, confirming probes within a tenant environment are not a viable option. The same results were obtained from Test #2 (Figure 31) with a probe located on the CSP network. Again, this probe was unable to capture any ping traffic, even though the pings were successful. This outcome was expected as the role of virtual and physical switches are to restrict broadcast of traffic. This natural switch behaviour can be overriden using SPAN ports, however traffic mirroring provides disadvantages such as duplication of traffic and other issues

outlined in section 5.4, above. Higher traffic visibility was obtained with a probe installed directly within the hypervisor of the server, see Test #3 (Figure 32), as the probe collected traffic passing directly through the vswitch installed within the server hypervisor environment. However, traffic visibility was restricted to just the server with the probe installed, with reduced visibility of traffic passing through other network devices. Test #5 (Figure 34) obtained the highest infrastrucuture visibility total of 100. However, as Figure 34 shows, this test required four probes to be positioned across the network. Restricting probes to the host server only, and excluding probes on cloud infrastructure devices such as routers, switches and firewalls, as in Test #4 (Figure 33), produced an almost identical infrastrucuture visibility total of 99. Although, locating the probes within the servers did restrict the traffic visibility across other CSP devices, which may impact the tracking of a bot as it propagates internally, such as to attack the storage infrastructure.

The purpose of the probe positioning tests was to empircally determine the optimum location of the IPFIX probes, defined above as the maximum traffic visibility for the minimal number of probes. Therefore the optimum probe positioning was in Test #4, with probes installed alongside the virtual switch within each server that hosts guest VMs. Whilst Test #4 does not provide complete infrastructure coverage, this placement selection should provide sufficient coverage to detect botnet communication between guest VMs or between tenants. If full visibility is required, Table 26 suggests the placement of an IPFIX probe on each critical CSP network device that requires protection. Albeit this configuration comes with two drawbacks; a) a complexity overhead of having to correlate data from multiple probes, and b) as the number of probes increases, so does the attack surface available to an adversary.

The empirical evidence provided from these tests back up *Johnston, et al.,* (2016) who analysed ways to incorporate a NIDS within Xen. They took a theoretical look at four potential locales to site the NIDS; (1) on the virtual network itself; (2) on the Netback driver (incoming traffic into Dom0); (3) on the Netfront drive (incoming traffic into DomU); and (4) on the hypervisor itself. They found that drivers could be installed that allow packets to be copied between unprivileged guest domains, meaning VM to VM traffic could be completely missed by a NIDS on virtual network, on the Netback driver or on the Netfront driver. In agreement with this research they propose the optimum location with maximum visibility for a NIDS is on the hypervisor itself.

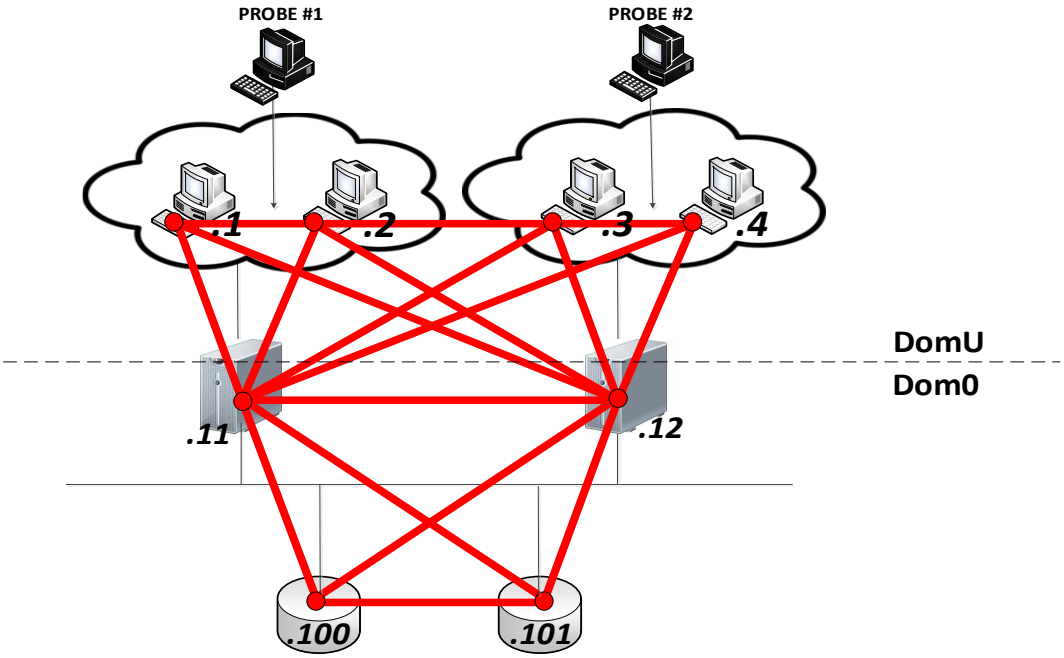


Figure 30. Location Test #1 - ICMP ping traffic captured by a probe in each tenant VM where red signifies a successful ping that was not captured by the probes.

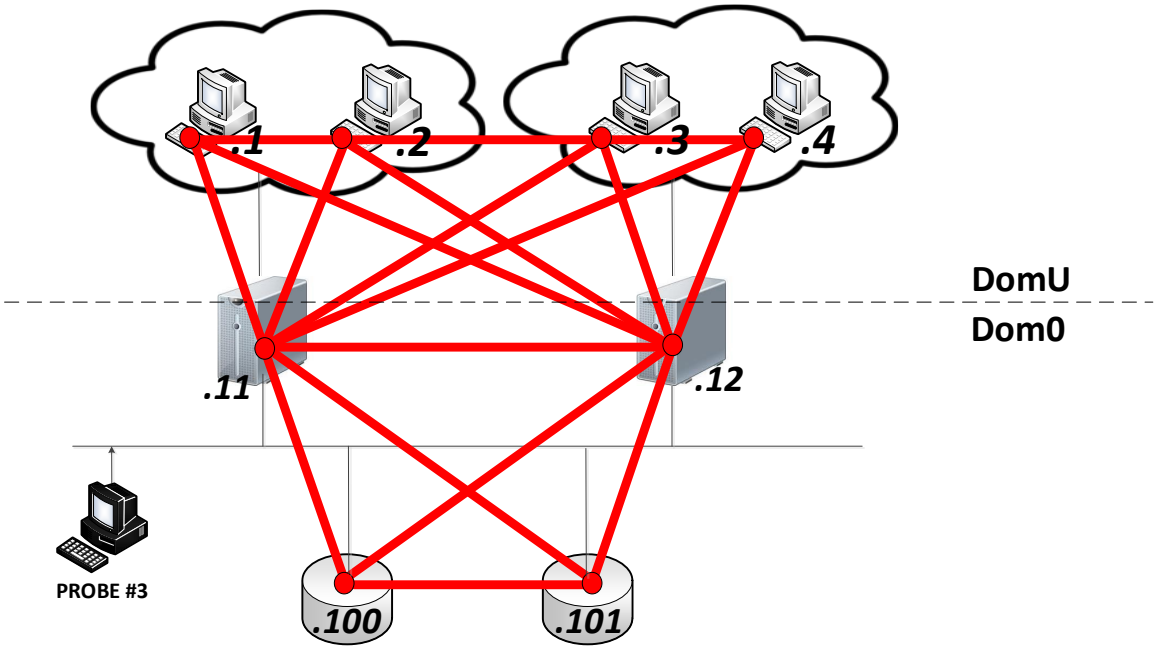


Figure 31. Location Test #2 - ICMP ping traffic captured by a probe on the LAN where red signifies a successful ping that was not captured by the probe.

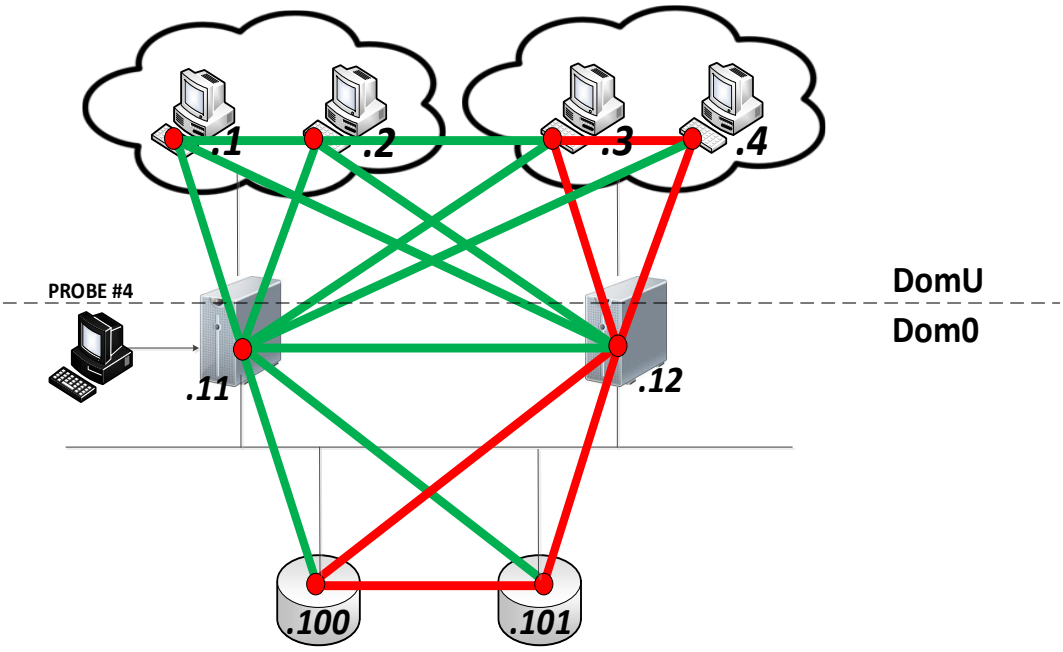


Figure 32. Location Test #3 - ICMP ping traffic captured by a probe on a host server where red signifies a successful ping that was not captured by the probe and green signifies a successful ping that was captured by the probe.

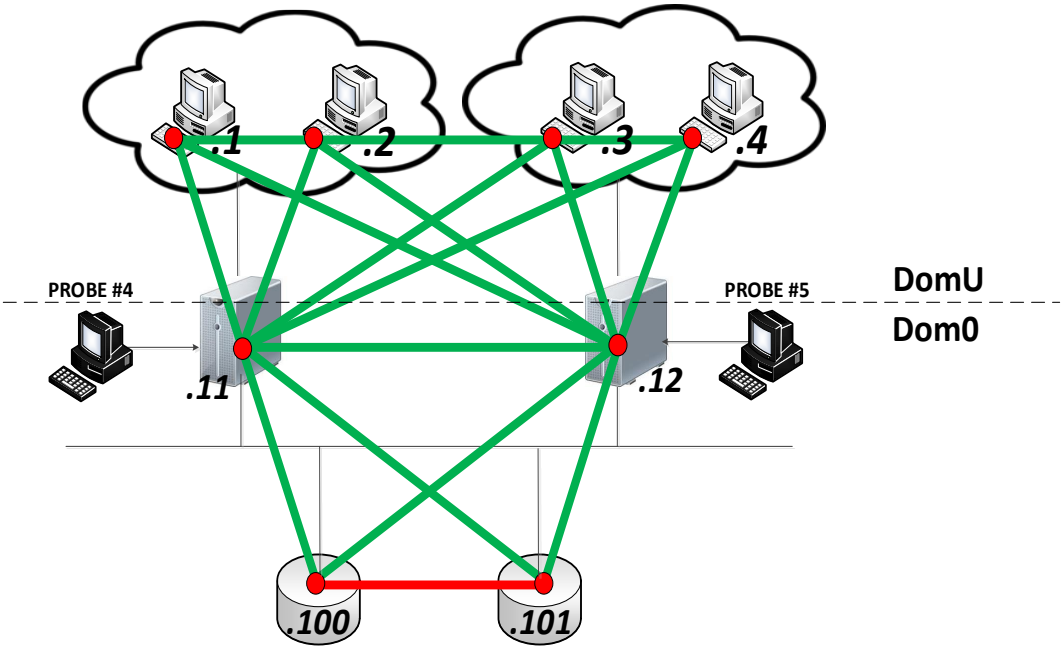


Figure 33. Location Test #4 - ICMP ping traffic captured by a probe on each host server where red signifies a successful ping that was not captured by the probes and green signifies a successful ping that was captured by the probes.

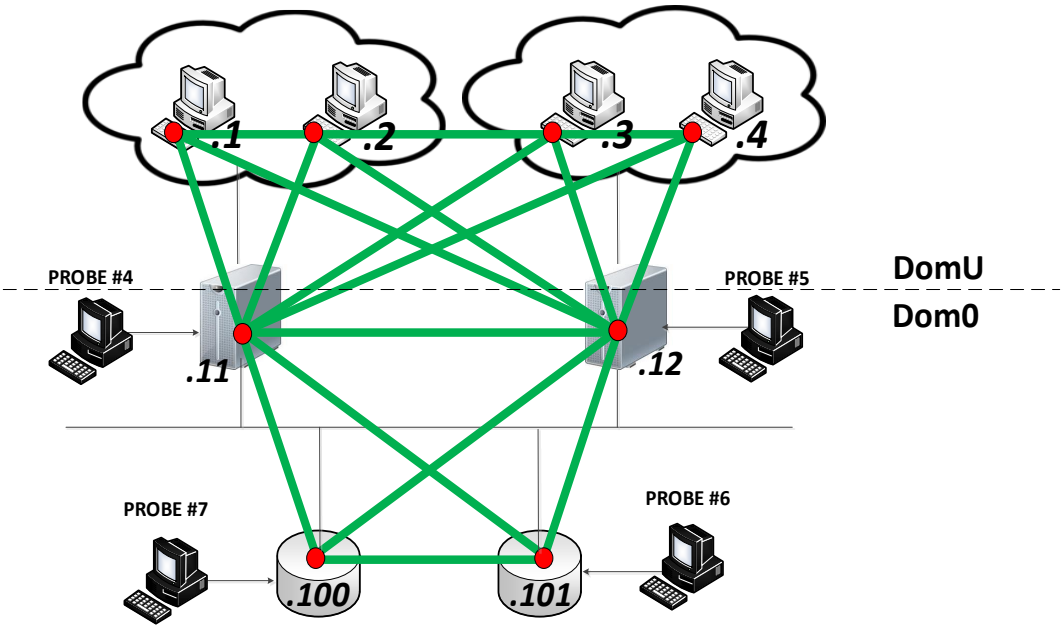


Figure 34. Location Test #5 - ICMP ping traffic captured by a probe on each device where green signifies a successful ping that was captured by the probes.

5.6 Probe Timing Test Methodology

In reconstructing the propagation of a bot across a network over time, the clocks of the collection devices distributed throughout a network must be synchronised to allow correlation of data flows by timestamps. In flow export, data packets are collected off the wire at an Observation Point, such as a device network interface. It is at this observation point, (i.e. the probe) that packets are pre-processed; which can include time-stamping as well as any data manipulation such as aggregation, sampling or filtering. One enhancement of IPFIX over NetFlow is the inclusion of security mechanisms such as SCTP, as detailed in chapter 3. SCTP ensures that IPFIX templates are sent reliably by improving end-to-end delay and count dropped packets; whilst PR-SCTP (partial reliability SCTP) add a mechanism to skip packet retransmissions (*Santos, 2016*). SCTP provides integrity of the flow data, mitigating against tampering attacks and packet insertion attacks.

If malware can impact data collection timestamps, it could successfully confuse the reconstruction of time related events. SCTP does little to protect against these sorts of timing attacks. Besides nefarious attacks, there may be occurrences in the cloud when guest VM timers are out of synchronisation due to misalignment, clock drift or global distribution across various time zones. Indeed, during testing of BotStack, it was found that even though steps were taken during configuration to attempt the consistent setting of clocks, some devices still had differing timers.

YAF uses the Libpcap library to timestamp each flow start and end time. Libpcap itself is reliant on the underlying OS kernel clock. The following set of tests alter clock timers on collection devices, to measure impact upon flow data timings across various probes, in order to understand if NTP (Network Time Protocol) can protect device synchronisation.

5.6.1 DATASET

As with the previous test, the dataset from this test was generated by systematically polling probe devices using ICMP ping traffic. After which, the clock settings were manually adjusted on either the servers or in the VMs, before repeating the test.

5.6.2 EQUIPMENT

The same test network, used in the previous probe placement tests, was used to test probe timings. VM#1, on Server #1, was assigned 192.168.0.1. VM#2, on Server #2, was assigned 192.168.0.4. VM#1 pinged VM#2 so that traffic must travel over two separate servers with probes attached, as indicated by the green path in Figure 35.

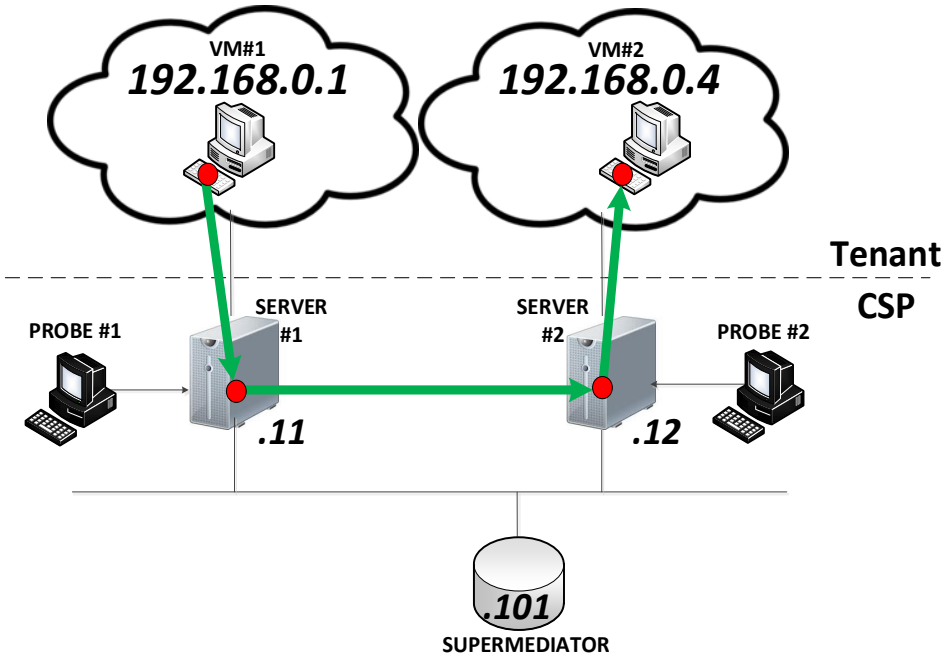


Figure 35. The logical architecture for the probe timing test environment where green indicates the path taken by the ICMP ping traffic.

Four clock alignment scenarios were tested:

- Server #1 and server #2 were both manually set to GMT, but not synchronised to NTP;
- Server #1 and server #2 were both set to GMT and synchronised to NTP;
- Server #1 is set to GMT, Server #2 was set to GMT +7. Both synchronised via NTP;
- Server #1 and server #2 were synchronised to GMT via NTP, whilst VM#1 and VM#2 were both manually allocated varying clock settings.

5.6.3 METHOD

The aim of this test was to understand how differences in device clock times impact the ability to coherently understand device data output from a holistic view. The independent variables were ICMP ping tests between devices. The dependent variables were the varying clock settings under test. Flow diagram Figure 36 summarises the testing method.

The detailed probe timer testing method for all four test scenarios was:

(1) The clock timers on devices were set as per the test requirement. Where NTP was required, clocks were synchronised to: `ntp.anglia.ac.uk`

(2) YAF was configured on the servers:

```
Server #1:     # yaf --live pcap --in xenbr0 --out timer_1.yaf
Server #2:     # yaf --live pcap --in xenbr0 --out timer_2.yaf
```

(3) Four ICMP pings were sent from VM #1 to VM #2

```
VM #1:      # ping 192.168.0.4 -n 4
```

(4) & (5) Step 3 was repeated 5 times in total, every 60 seconds

(6) YAF was stopped;

(7) The resulting `.yaf` files were saved for later analysis.

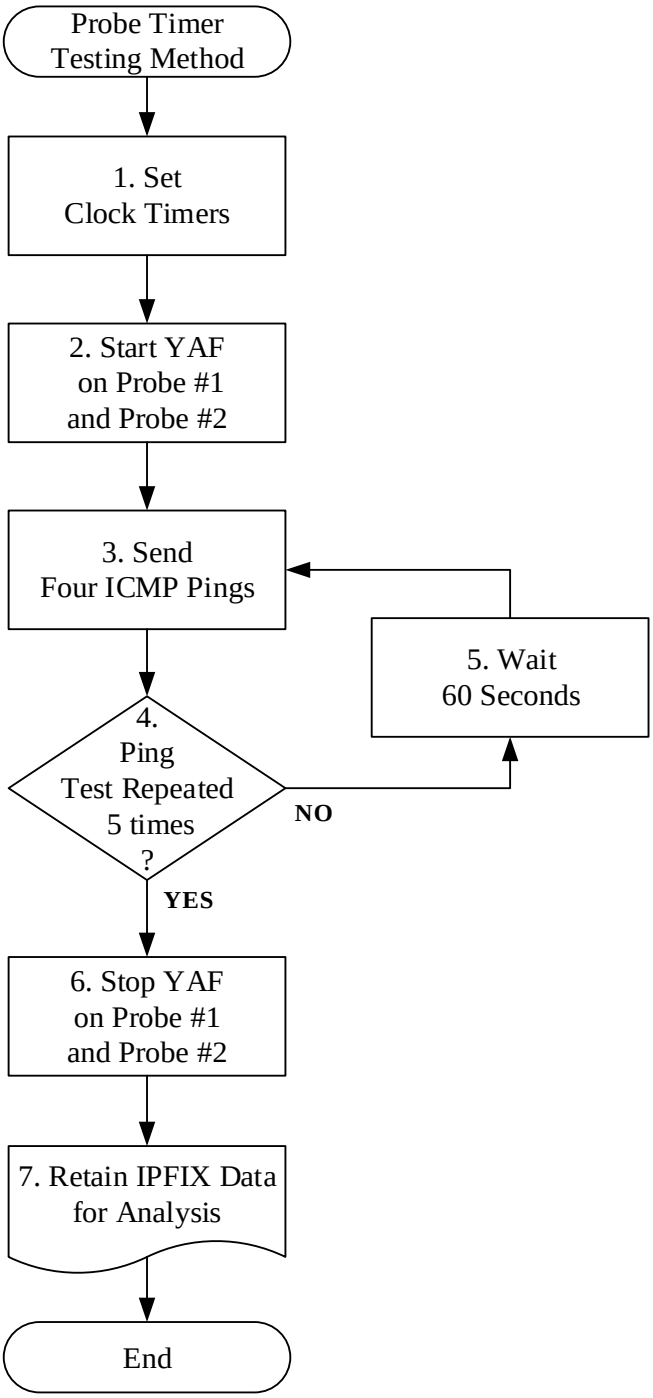


Figure 36. Flow diagram of the probe timer misalignment test.

5.6.4 ANALYSIS

To understand the impact of clock time misalignment, the timestamps of the flow start times were manually compared between probes on server #1 and server #2 during each timing test.

5.7 Probe Timing Results

The results of the ping tests results for the four scenarios are provided in Tables 27 - 30. These tables showed ICMP ping traffic only, with all other network traffic removed. In each of the four tests, four pings were sent five times, every 60 seconds. For clarity of result presentation, sufficient ping traffic is detailed in order to show the difference in timing, with other subsequent pings being removed. Additionally, flow start times have been converted to EPOCH times for ease of readability.

During the four tests, ICMP pings were manually issued via the command line. It was not always possible to precisely time 60 seconds between each ping, hence not every batch of pings occurs exactly 60 seconds after the previous batch. Results in Table 30 offer an example of this, with 1 minute and 2 seconds between manual pings. Minimal inconsistencies in ping timing initiation is not anticipated to impact test results.

TABLE 27. TIMING TEST #1 - BOTH SERVER CLOCKS MANUALLY SET TO GMT

collector	sIP	dIP	packets	protocol	sTime	ping
Probe #1	192.168.0.1	192.168.0.4	2	icmp	08/19/2015 14:41:19	1
Probe #2	192.168.0.1	192.168.0.4	2	icmp	08/19/2015 14:41:19	1
Probe #2	192.168.0.1	192.168.0.4	4	icmp	08/19/2015 14:42:18	2
Probe #2	192.168.0.4	192.168.0.1	4	icmp	08/19/2015 14:42:18	2
Probe #1	192.168.0.1	192.168.0.4	2	icmp	08/19/2015 14:42:19	2
Probe #1	192.168.0.4	192.168.0.1	2	icmp	08/19/2015 14:42:19	2
Probe #1	192.168.0.1	192.168.0.4	2	icmp	08/19/2015 14:43:10	3
Probe #1	192.168.0.4	192.168.0.1	2	icmp	08/19/2015 14:43:10	3

Note how ping #3 impacts ping #2 due to timestamp inconsistencies.

TABLE 28. TIMING TEST #2 - BOTH SERVER CLOCKS ARE SYNCHRONISED TO GMT, VIA NTP

collector	sIP	dIP	packets	protocol	sTime	ping
Probe #1	192.168.0.1	192.168.0.4	4	icmp	08/20/2015 12:38:19	1
Probe #2	192.168.0.4	192.168.0.1	4	icmp	08/20/2015 12:38:19	1
Probe #1	192.168.0.1	192.168.0.4	4	icmp	08/20/2015 12:39:19	2
Probe #2	192.168.0.4	192.168.0.1	4	icmp	08/20/2015 12:39:19	2

Note how the ICMP pings are now synchronised and aggregated.

TABLE 29. TIMING TEST #3 - SERVER #1 SET TO GMT, SERVER #2 SET TO GMT +7, BOTH VIA NTP

collector	sIP	dIP	packets	protocol	sTime	ping
Probe #1	192.168.0.1	192.168.0.4	4	icmp	08/20/2015 13:55:44	1
Probe #2	192.168.0.4	192.168.0.1	4	icmp	08/20/2015 13:55:44	1
Probe #1	192.168.0.1	192.168.0.4	4	icmp	08/20/2015 12:56:46	2
Probe #2	192.168.0.4	192.168.0.1	4	icmp	08/20/2015 12:56:46	2

Note that whilst server #2 is set 7 hours in front of server #1, the time zone difference is not transferred to the ICMP timestamps.

TABLE 30. TIMING TEST #4 - BOTH SERVERS ARE SYNCHRONISED TO GMT, VIA NTP
BOTH VMS ARE MANUALLY ALLOCATED DIFFERENT TIMES

collector	sIP	dIP	packets	protocol	sTime	ping
Probe #1	192.168.0.1	192.168.0.4	4	icmp	08/20/2015 14:23:59	1
Probe #2	192.168.0.4	192.168.0.1	4	icmp	08/20/2015 14:23:59	1
Probe #1	192.168.0.1	192.168.0.4	4	icmp	08/20/2015 14:25:01	2
Probe #2	192.168.0.4	192.168.0.1	4	icmp	08/20/2015 14:25:01	2

Note how the different VM clock settings are not transferred to the ICMP timestamps.

5.8 Discussion

The IPFIX framework proposed in Table 24 comprised of individual open source technology elements. The justification for the selection of each framework element is detailed above. BotStack was designed to be modular, in that each element can be replaced by another element where necessary. For example, a CSP may have an existing relationship with a flow collection partner such as Plixer, who provide closed source incident response forensics analysis software which integrates with IPFIX. SuperMediator could be replaced with Plixer's Scrutinizer software because IPFIX is a ratified standard. It was discussed above, that the only open source IPFIX collectors and exporters that are truly customisable are YAF and nProbe. A limitation of YAF is that it is only supported on Linux. Again, a standards based approach to IPFIX allows YAF to be replaced with nProbe, although not all YAF EE's are available in the nProbe template. Another limitation of YAF is the lack of documentation around how to create customised EEs. This is also true of nProbe. It should be noted if an element in BotStack is replaced, interoperability cannot be confirmed, as neither Scrutinizer nor nProbe were tested with the framework elements. There is scope for creation of an open source IPFIX exporter/collector pair that fully support the construction of new EEs into an IPFIX template

The open source nature of BotStack makes it a practical test bed for academic research. BotStack was constructed upon the premise of allowing IPFIX export to be incorporated into CSP environments built upon an IaaS model. The flexibility provided by the ability to replace framework elements allows BotStack to be ported to other cloud service models that are built upon virtualised infrastructure, such as Platform as a Service (PaaS) or Storage as a Service (SaaS).

Evidence is provided in Chapter 5.5 that siting IPFIX probes upon virtual switches within servers hosting guest VMs, provides optimal probe location in detecting botnet propagation between tenanted environments. In a Xen environment, such as BotStack, which utilises para-virtualisation, this means an IPFIX probe should be installed within the hypervisor domain. Where full CSP infrastructure protection is required, an IPFIX probe should be installed in each critical network device that requires protection. Further work is needed to understand probe positioning in non-Xen hypervisors that do not support para-virtualisation.

Security mandates within RFC-7011 (*Internet Engineering Task Force, 2013a*) should mitigate most of the known techniques by which malware can influence IPFIX collection. IPFIX flows can be obfuscated via TLS encryption, mitigating packet inspection and tampering, whilst SCTP mitigates against replay attacks and DDoS. Note that whilst YAF supports both TLS and SCTP, testing these features is beyond the scope of this thesis, as it is the manufacturer's responsibility to provide functionality against the IPFIX standard. However, malware continues to evolve to take advantage of new attack vectors. One potential technique for future malware to evade detection is to tamper with the timestamps of collected data.

Tables 27-30 provide evidence that BotStack should be immune to malware attempting timer misalignment attacks. This is primarily because the tenanted environments take their clock timings from the host servers. If guest VM clocks are manually set so as to be different to the host server clocks, the timestamps on traffic captured by the probe is not impacted because probe packet timestamps are also taken from the server (Table 30). Hence, an attacker could not attempt to mis-align probe capture timings by amending clocks in a guest VM. Likewise, the probes are not impacted by time zone variations. When server clocks are set to different time zones, the difference between clocks in different zone are not transferred to the packet timestamp (Table 29). This means that probes can be distributed across a global network and export into a single data collection point without need to normalize clocks before data analysis. However, server clocks should be synchronized via NTP in order to ensure probe timing consistency. When clocks across multiple servers are

manually configured to be synchronization as close as possible, a small discrepancy in clock times can impact the perceived probe capture timings. Table 27 shows how ICMP ping traffic timestamps can be mis-interpreted by IPFIX aggregation engines when server clocks are set manually, but this is rectified when server clocks are set via NTP (Table 28). The impact of an attack on the NTP protocol itself was not tested. A method for future malware to perform a timing attack would be to physically disable NTP on the host device and force desynchronisation upon the host clocks. This could be an additional device that periodically checks probe hosts for clock synchronisation. A limitation of NTP is it only timestamps to millisecond precision. Precision Time Protocol (PTP) is accurate to 100 nanoseconds. Whilst IPFIX supports capture at nanosecond granularity (IANA_ID#156 and IANA_ID#157) no literature could be found evaluating IPFIX nanosecond time-stamping. The expense of PTP enabled hardware placed the testing of PTP beyond the scope of this study.

5.9 Summary

Chapter 1 identified the gap in the knowledge in the understanding of how the next-generation of flow protocols, such as IPFIX, can be applied to botnet detection. Having created two IPFIX templates for botnet capture in chapter 4, this chapter made several contributions by addressing research objective #3; the construction of an IPFIX export framework that enables the capture of botnet communication traffic across cloud provider networks.

The first contribution from this chapter was BotStack, a novel IPFIX framework for CSPs. Each BotStack element is open source, with justification for inclusion provided. However, the flexibility of BotStack permits each element to be replaced with an alternative element should a CSP have a preferred vendor relationship, hence easing the migration process from the CSP's current environment to one that supports IPFIX. Whilst BotStack was constructed for an IaaS model, the framework is flexible enough to be incorporated into PaaS, SaaS or IoT type models.

The second contribution was empirical evidence that the optimal siting of the IPFIX probes is within the hypervisor of tenant hosting servers. This presents many advantages. Removing the probe from the tenant environment not only negates any tenant surveillance concerns and respects tenant privacy, but also reduces the probe attack surfaces to malware housed within the tenant environment. A cost advantage is presented as this requires fewer probes for maximum traffic visibility. Additionally, a

management advantages arises as fewer probes reduces complexities in correlating multiple device data streams.

The third contribution was evidence that locating the IPFIX probe within the hypervisor reduces the risk of malware timing attacks. Whilst IPFIX security features mitigate risks to the IPFIX data stream, they do little to protect the probe itself from attack. Evidence shows that for as long as probes are synchronised via NTP, the ability for malware to influence the timestamp on the data flows is minimised.

The next chapter validates the interoperation of the BotStack framework created within this chapter, with the BotProbe templates constructed in chapter 4. This is achieved by deploying a real world botnet into a proof of concept network based on the IPFIX framework, and analysing the results of traffic captured via the IPFIX templates.

6
Concept Validation

6.1 Introduction

Chapter 4 described the creation of two novel IPFIX templates for capturing botnet communication traffic. Chapter 5 described the construction of an IPFIX framework which introduces IPFIX traffic capture into a cloud architecture. This chapter validates the interoperation of the BotProbe templates with the BotStack framework. A proof of concept test network, constructed from the BotStack framework, is infected with the Zeus botnet. BotProbe is used to capture botnet communication traffic, which is presented both as a visualisation of the overall attack process, and as four distinct profiles of a bot life cycle.

6.2 Botnet Life Cycle Model

Several scholars have defined the notion of a botnet's life cycle (*Feily, Shahrestani and Ramadass, 2009; Liu, et al., 2009; Govil, 2007; Gu, et al., 2007*). These models primarily describe early life cycle processes, such as victim infection and C&C server registration. However, a botnet generates collectable traffic throughout its entire life cycle. The defining moment in a botnet life cycle is the attack phase, before which detection and takedown should happen. As the most relevant model to this study, BotHunter (*Gu, et al., 2007*) is extended to include latter life cycle elements, thereby creating a four phase botnet life cycle model where each phase displays a distinct traffic profile:

- Scanning, where the bot scans for propagation options (*Gu, et al., 2007)*;
- Infection, when the victim downloads the bot, which then registers with the C&C server (*Gu, et al., 2007)*;
- Maintenance, including bot keep-alive beacons, software updates;
- Attack, includes the bot attack, such as DDoS, and post attack actions, such as transfer of stolen data back to the C&C server.

6.3 Proof of Concept Validation Methodology

6.3.1 DATASET

The experiment used the Zeus Crime-wave Toolkit v2.0.8.9. The toolkit includes the Zeus C&C panel software, allowing the researcher to retain full control of the bot across its entire life cycle. Zeus was first discovered in 2007 stealing banking information via a man-in-the-browser attack. Despite its age, new active Zeus C&C servers are still being discovered, making this a suitable target botnet as a test subject.

6.3.2 EQUIPMENT

A proof of concept test network was constructed using the BotStack components outlined in Table 24. The logical architecture for this network is described in Figure 37. Two Dell PowerEdge R710 servers, each with four Intel Xenon 5160 3.0GHz CPUs and 8GB RAM, ran Ubuntu 14.04 LTS desktop operating systems. Xen 4.4.0, the XAPI toolstack, Open vSwitch v2.0.2 and YAF v2.8.4 were installed into Dom0. Each server was configured with five VMs. Server #1 held IP addresses 192.168.0.11x and server #2 held 192.168.0.12x. Server #1 had a Windows XP Pro SP3 VM acting as the botmaster C&C server. This VM had Zeus Toolkit v2.0.8.9 installed, by which a bot.exe binary was created and configured to call back to the C&C server every 60 seconds. The remaining nine VMs were Windows 7 Pro SP1. XenCentre v6.5 was installed on an additional PC to manage the VMs. The network was configured as a flat 192.168.0.0/24 network.

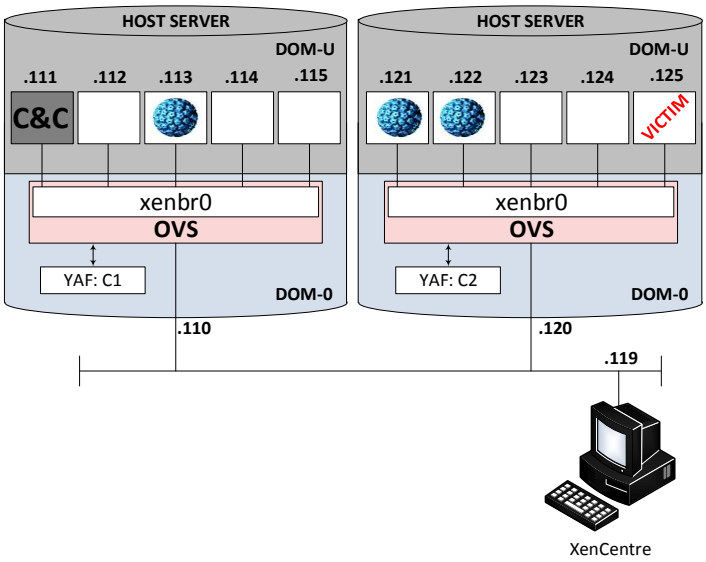


Figure 37. The logical architecture for the proof of concept network.

6.3.3 METHOD

The aim of this test was to provide demonstrable evidence of the inter-workings of the IPFIX framework and IPFIX template in capturing the four phases of a botnet life cycle. The dependent variables in this test were the test network and capture methods. The independent variables, which are the source codes of the bots, were constrained to a single bot (Zeus v2.0.8.9). The test could have been performed with a range of various bot C&C server types to compare traffic profile signatures. Figure 38 summarises the testing method. The detailed proof of concept testing method was:

(1) Each server was configured using a bash script

```
# bash startup.sh
```

The bash script (Appendix E) configured Open vSwitch with the appropriate VIF settings and synchronised the server clocks with an NTP clock on the Internet;

(2) Each of the 10 VMs were launched. ICMP ping tests confirmed connectivity;

(3) YAF was configured to export IPFIX to the local server:

```
# yaf --live pcap --in xenbr0 --out pocflow.yaf --rotate 60
-v --plugin-name=/usr/loca/lib/yaf/dpacketplugin.la
--applabel --max-payload 65535 --tls
```

- --applabel and --max-payload are both required for dpacketplugin.la;
- --rotate 60 = saves pocflow.yaf every 60 seconds;
- --tls = encrypts IPFIX traffic using TLS

(4) The *scanning phase* is simulated by the C&C VM using nmap in stealth mode:

```
# nmap -sS 192.168.0.112-115
```

(5) The *infection phase* is simulated by three VMs (192.168.0.113, 192.168.0.121, 192.168.0.122) downloading the bot.exe binary from the XP VM (192.168.0.111);

(6) The *maintenance phase* is simulated by leaving the network to run for a period of 10-20 minutes in order to generate bot keep-alive traffic;

(7) The *attack phase* simulated a DDoS attack from infected VMs upon a victim:

```
# ping -n 5000 -l 1500 192.168.0.125
```

(8) YAF was terminated to stop traffic capture;

(9) IPFIX templates (Appendix C) were applied to create data for analysis:

```
# super_mediator --config botprobe.conf
# super_mediator --config extended.conf
```

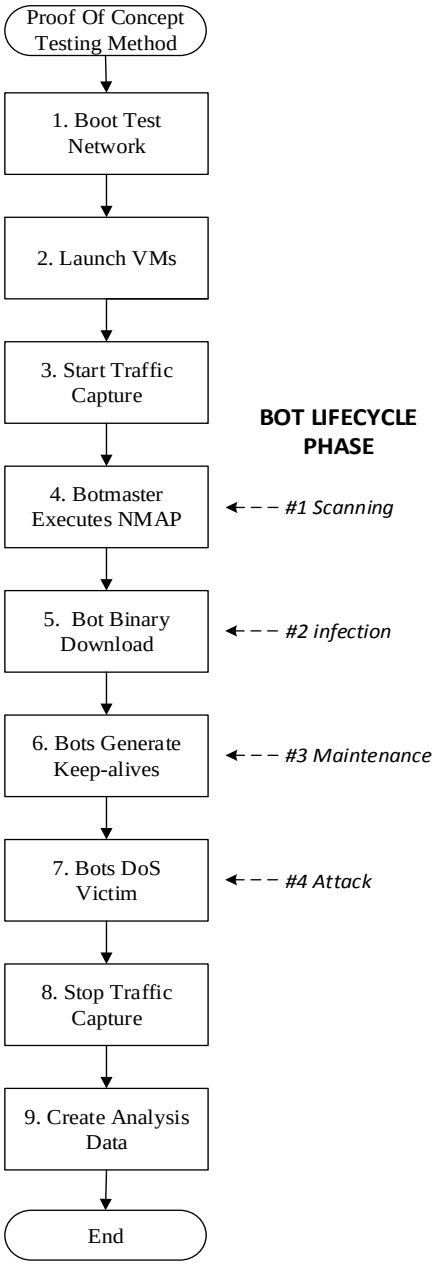


Figure 38. Flow diagram of the Zeus botnet traffic capture test.

Traffic capture in a live network meant that it was anticipated that background noise from Windows VMs and other networked devices may obfuscate bot traffic. To maintain a realistic scenario, no attempt was made to reduce or eliminate this background noise, as the background noise anticipated in this test will also be present in a CSP network. *Zhao, et al.,* (2013) suggested a realistic test environment can be created by introducing HTTP web browsing traffic, online gaming packets, bit torrent clients and email traffic. This was considered overkill for this proof of concept test.

The Zeus bot was under the control of the researcher throughout the test, via the Zeus C&C panel. This control over each bot activity meant that the bot's behaviour

was predictable and consistent each time a test was performed. Therefore, the proof of concept test method was only performed once.

6.3.4 ANALYSIS

To confirm the interoperability between the IPFIX framework and IPFIX template, the exported IPFIX traffic was manually interpreted to identify the four phases of Zeus bot life cycle over time; scanning, infection, maintenance and attack. The overall life cycle timeline was visually displayed as an arc diagram. Each of the four life cycle model profiles were visually displayed as individual property graphs of each phase.

6.4 Proof of Concept Results

The exported IPFIX data was passed through SuperMediator using both the BotProbe and extended BotProbe templates. When the entire captured IPFIX dataset is plotted over time, see Figure 39, the phases of the life cycle are visible. The arc diagram displays the C&C server (.111) scanning the network via TCP SYN, the HTTP traffic between the three VMs (.113, .121 and .122) and the C&C server (.111) in the infection phase, followed by considerable ICMP ping traffic between the three VMs and a victim (.125) during the attack. The behaviour displayed in this diagram indicates that the VMs downloaded a botnet binary from the C&C server which was then used to attack the victim. The BotProbe template yields an identical arc diagram as the extended BotProbe template, as both templates feed from same captured IPFIX dataset. The extended BotProbe template providing more application layer detail, which the BotProbe template uses port numbers to identify the application.

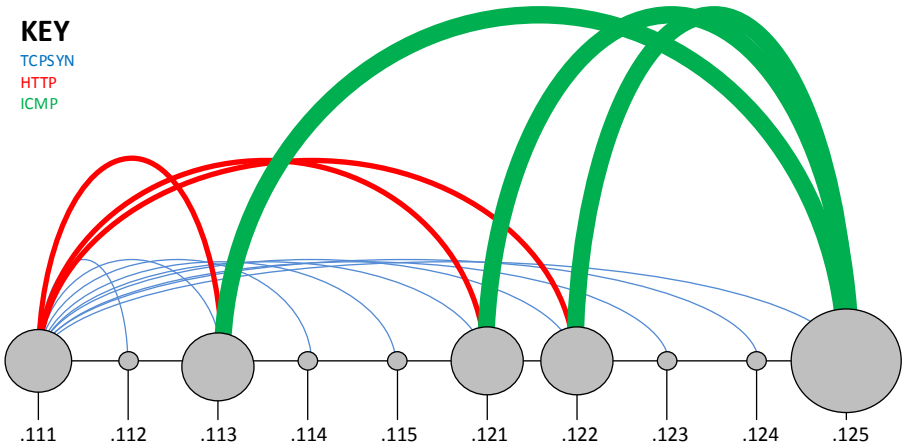
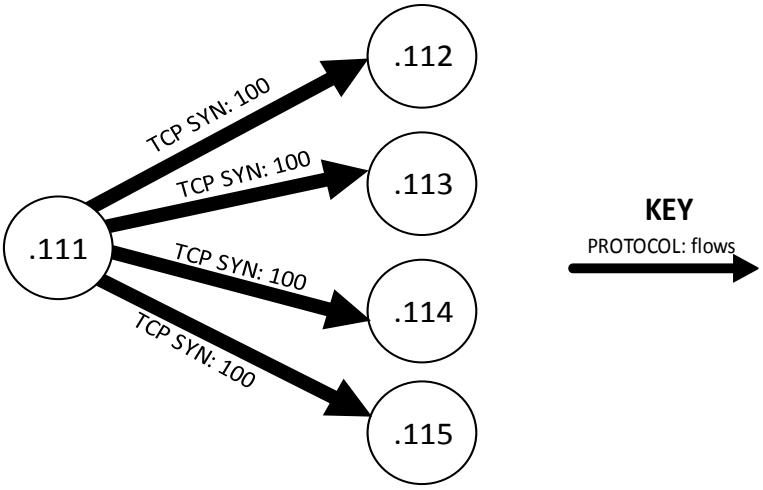
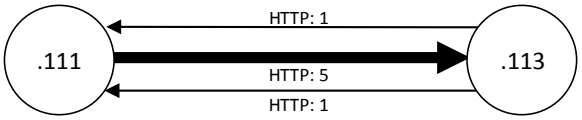


Figure 39. Arc diagram of the entire botnet infection.

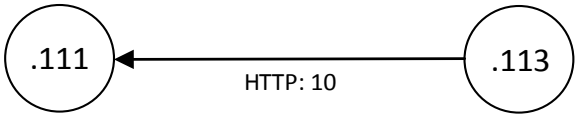
The property graphs in Figure 40 were created through manual interpretation of the exported IPFIX data, in order to visually display the traffic profiles across each bot life cycle phase. Figures 39 and 40 are discussed in Chapter 6.5 below.



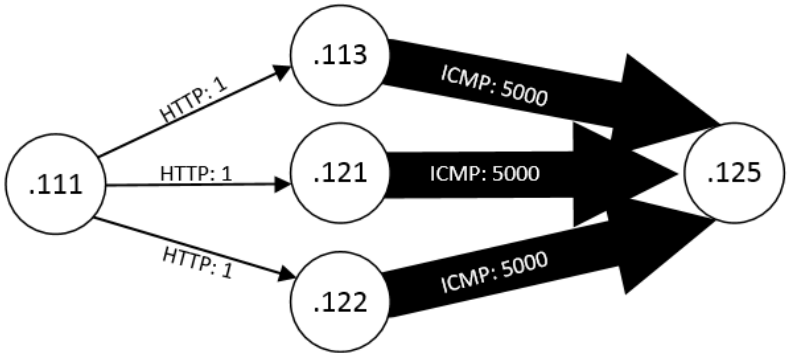
Scanning - timeframe 15:00:01 - 15:10:30.



Infection - timeframe 15:15:01 - 15:16:03.



Maintenance - timeframe 15:31:00 - 15:41:00.



Attack - timeframe: 15:45:01 - 15:47:03.

Figure 40. Four botnet life cycle profiles visualised through IPFIX export data.

6.5 Discussion

Bot authors generally concentrate their efforts on development of the bot at a code level; getting a bot to perform a new task, to employ better obfuscation methods, or to utilize a new communications channel protocol. However, fundamental botnet communication characteristics have remained unchanged.

BLINC, (*Karagianis, Papagiannaki and Faloutsos, 2005*) generated a library of traffic profile signatures using visual representation of traffic profiles, by plotting srcIPv4 and dstIPv4 for various applications. This library describes many network applications, but unfortunately omits botnets profile signatures.

The property graphs in Figure 40 were created using a similar method to BLINC, by manually plotting the nodal communications from the exported IPFIX capture of botnet traffic. Each node represents an IP address on the network, with the edges representing the size and direction of traffic communication, typically by protocol. When considering the nodal and edge distribution over time it is possible to observe the four distinct traffic profiles of a botnet attack lifecycle, as describes in Chapter 6.2 above.

The scanning phase (Figure 40, top) is clearly depicted via short bursts of TCP SYN traffic from a single node (the C&C server in VM .111) to multiple IP addresses. This matches the expected traffic profiles as expected from the NMAP command issued during the tests. The infection phase (Figure 40, upper middle) depicts the C&C server infecting a victim VM .113, with the victim reporting back to the C&C once the botnet .exe has been installed. Likewise, with the maintenance phase (Figure 40, lower middle), short HTTP update packets were seen from the victim to the server once a minute over the 10 minute testing phase. The attack phase (Figure 40, bottom) shows, over a period of two minutes, the C&C issuing attack commands to three bots which perform an ICMP denial of service on a victim. This compares well with the attack signature created by BLINC (*Karagianis, Papagiannaki and Faloutsos, 2005*), as shown in Figure 41 below. BLINC shows a single host scanning an address space to identify vulnerabilities, followed by an attack on a single destination port. The difference between Figure 41 and Figure 40, is that the volume of ICMP DOS traffic in Figure 40 is represented by the thickness of the edge between the nodes.

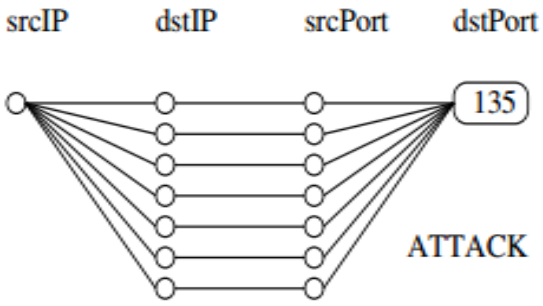


Figure 41. Traffic scanning and attack profile.
(*Karagiannis, Papagiannaki and Faloutsos, 2005*)

The arc diagram in Figure 39 is an alternative interpretation of the IPFIX export data collected from the botnet attack. Compared with Figure 40 which shows the four attack phases as independent events over time, Figure 39 displays the entire timeline of the infection in one diagram. A cursory glance at the traffic profiles in Figure 39 shows what could be "normal" peer-to-peer network traffic between a server (node .111) and clients. It is only when protocol information captured from the packet header using the BotProbe template, is overlaid onto the arc diagram, that strong evidence of anomalous traffic becomes apparent. Protocol information in the arc diagram suggests that one VM (.111), which is not a server, contacts another three VMs (.113, .121 and .122) via HTTP. After which, these three VMs send high volumes of ICMP data to a fifth VM (.125). This traffic could be a legitimate file transfer over HTTP, although the ICMP ping traffic to the victim VM suggests suspicious activity. This activity can be confirmed as malicious when more detailed HTTP information, captured via the extended BotProbe template, is applied. Analysis of the HTTP requests show the victim requesting the configuration files and bot executable from the C&C server:

```
GET /cfg1.bin HTTP/1.1

POST /index.php HTTP/1.1

GET /bot.exe HTTP/1.1
```

This GET/POST conversation is as is expected from the basic set up of Zeus bot in the proof of concept test. This corroborates with the findings of *Binsalleeh et al,* (2010) and *Falliere and Chien,* (2009) in their dissection of the HTTP Zeus botnet.

The full opportunity for applying IPFIX data as a feeder mechanism into botnet detection using graph theory has not yet been explored. *Collins and Reiter* (2007)

demonstrated how a bot impacts a graph's structure by connecting otherwise unconnected components. *Françios, et al.,* (2011) feed weightings based on nodal neighbours into a PageRank algorithm to detect botnets. Their work is based on number of neighbours, as opposed to any specific traffic attributes. Botyacc, (*Nagaraja, 2014*) used separated spatial and temporal metrics in property graphs to separate benign from malicious P2P traffic. IPFIX provides several mechanisms to extend the study of graph-based botnet detection. IPFIX is able to capture of any number of yet unstudied traffic attributes and characteristics, of which some may be malicious activity indicators.

IPFIX exports data in a highly structured format, meaning IPFIX can be easily applied to two new avenues of botnet research. Structured data makes it easier to feed IPFIX export data directly into graph database systems for visualisation and data querying of botnet propagation. Work is underway to channel the feeds from multiple IPFIX probes distributed across a network into graph analysis software such as neo4j[8], with which it should be possible to visualise the bot life cycle as the bot propagates across a network in real-time. Additionally, as IPFIX data is highly structured, it makes it more suited, than unstructured packet capture traffic, to feed into detection engines. This provides possibilities for new machine AI engines that can be trained to recognise patterns. One such challenge in botnet detection is the issue of application beaconing. Beaconing impacts the number of false-positive events in botnet detection as a legitimate application behaves in a similar way to a botnet C&C, with regular keep-alive traffic between the server and client. The combination of IPFIX with graph theory may open up new areas of research for AI to distinguish legitimate application beaconing to malicious botnet updates, particularly when post-beaconing events are included in the analysis. Whilst an AI can be trained to recognise patterns, in security data analysis a human mind is still required to interpret the patterns found within (*Collins, 2014*). These are discussed in more detail in Chapter 7 - Future Work.

6.6 Summary

Chapter 1 identified a gap in the knowledge in the understanding of how the IPFIX protocol can be applied to botnet detection within CSP infrastructures. The contribution from this chapter came in addressing research objective #4 in validating the effectiveness of BotProbe and BotStack as a botnet traffic capture mechanism that can be deployed within a typical CSP environment. The results of data analysis in

[8] https://neo4j.com/

Figure 40 confirmed that botnet traffic was captured across each phase of a four phase bot life cycle: scanning, infection, maintenance and attack. The traffic profile results achieved by the BotProbe template were comparable with the NetFlow v5 findings of BLINC (*Karagiannis, Papagiannaki and Faloutsos, 2005*). The key difference being that each NetFlow v5 flow captured by BLINC was 48 bytes in length, compared with the BotProbe template which was 43 bytes and contains considerably more information. This chapter has also demonstrated how suspect traffic can be more rigorously interrogated at the application layer using the extended BotProbe template. Whilst data from the BotProbe template is sufficient to insinuate the presence of a botnet in one of more of the four life cycle phases, capture of HTTP GET statements in the extended BotProbe template provide further confirmation that this traffic is indeed of bot origin. This provides evidence to the argument that CSPs could provide a higher level of threat detection should tenants be willing to disclose a minimal amount of application/payload data.

The test undertaken in this chapter was performed as a proof of concept, in a cloud environment. Whilst the IPFIX framework was constructed from open source technologies that are common in a cloud infrastructure, the framework does not cover all technology possibilities. The framework is modular, providing flexibility to replace certain aspects, thus providing a platform to test IPFIX in various environments; such as different hypervisors, vswitches or IPFIX exporter/collector pairs. Furthermore, EEs bring flexibility to IPFIX template construction allowing the testing of new attributes that may assist in botnet detection.

The final chapter draws conclusions from all the evidence collated within this thesis, indicating areas for further development.

7 Conclusions

The first chapter in this thesis outlined a gap in the understanding of how IPFIX export can be applied to botnet traffic capture. The IPFIX protocol, ratified as RFC-7011 through RFC-7015 in 2013, is the present standard for flow export. As IPFIX was specifically designed to overcome weaknesses in the much older NetFlow export protocol, a hypothesis was presented that IPFIX should offer clear advantages over NetFlow. The originality of this work came from investigating how IPFIX is superior to NetFlow in the construction of a botnet capture mechanism. Evidence of similar investigation by other researchers prior to this study could not be found.

The motivation to move this study into a cloud service provider environment arose as more organisations and individuals opt to outsource some, or all, of their IT requirements to the cloud; be that for storage, on-demand processing power or for cloud-hosted software services. These three factors are all contributors toward the cloud becoming an important element of the Internet of Things and smart city area networks. Traditional signature-based malware detection systems do little to protect these areas from botnet attack.

This final chapters states how the evidence collated throughout this research project demonstrates real advantages of the IPFIX protocol over NetFlow for a novel methodological approach to botnet traffic capture in cloud service providers. This is presented through four original contributions to knowledge. Limitations of the study are then considered, along with various areas for future study in the application of IPFIX to botnet detection. This thesis closes with concluding remarks on the impact and importance of this work.

7.1 Contributions to Knowledge

This study set four research objectives to answer the hypothesis that IPFIX offers advantages over NetFlow v5 for botnet communication traffic capture in a CSP environment. Through evidence provided in earlier chapters towards validating this hypothesis, four original contributions to knowledge are made, as detailed below.

Contribution #1 - Evidence from a critical investigation into IPFIX state of the art, to suggest that the design of the IPFIX export protocol has advantages over NetFlow v5, when applied to botnet communication traffic capture in cloud provider networks.

Addressing research objective #1, chapter 2 investigated the risk of botnet attacks on cloud service providers, arguing that internal cloud infrastructure is vulnerable to attack; in particular storage or co-resident tenants. This attack vector occurs mainly due to software vulnerabilities in hypervisors. The Crisis malware (*Katsuki, 2012*) and Venom (*CVE-2015-2456*) being two such examples. *Dillion and Winters* (2014) considered how trends to offload network edge-device intelligence to the cloud will allow devices and sensors to be built that require lower CPU capability and thereby reducing power consumption. In 2016, the Mirai botnet demonstrated the potential of damage from IoT hosted botnets (*Mansfield-Devine, 2016*). The impact of bot attacks is not just limited to CSPs. As home networks increase, TVs, thermostats, smoke alarms, and Internet connected white goods such as toasters and fridges, all become an attack surface for botnets. The future will see cloud centralised data storage as an essential building block in the IoTs. This will increase the likelihood of attacks upon CSPs. Much of this analysis in Chapter 2 has been presented at CFET 2014 (*Graham and Winckles, 2014; Graham, Winckles and Moore, 2014*) and OWASP 2014 (*Graham, 2014*).

Scholars have known that NetFlow is limited in its application to threat detection *(Velan, 2013; Gates, et al., 2004),* proposing that IPFIX will become a superior in next-generation networking *(Velan, Jirsik and Čeleda, 2013).* With cloud provider attack vectors mapped to chapter 2, chapter 3 compared the design enhancements of IPFIX with NetFlow v5 and NetFlow v9. This critical investigation revealed seven areas where IPFIX offers direct advantage over NetFlow. When these advantages are applied to the creation of a botnet traffic capture mechanism for a cloud provider, the standards-based approach of IPFIX becomes important to ensure vendor interoperability. The impact of a standards-based approach is highlighted in Chapter 4,

where the BotProbe template is constructed to work with the YAF IPFIX exporter, but standardisation allows this template to be ported to the nProbe IPFIX exporter.

Another key advantage of IPFIX is template customisation. Chapter 3 argues that template customisation overcomes the rigidity of the fixed NetFlow v5 template. Support in IPFIX for enterprise element creation and variable length fields should allow the efficient capture of botnet application layer traffic attributes, such as HTTP GET statements. A condensed version of the comparison between IPFIX and NetFlow was presented at BotConf 2015 (*Graham, Winckles and Sanchez, 2015b*).

During the course of this research, it became evident that IPFIX can be applied to multiple threat detection scenarios. Work is underway to create IPFIX templates to capture spam traffic and malicious HTTP traffic. This granularity for defining and controlling attribute capture opens up IPFIX traffic to analysis through machine learning, which ultimately leads to automation of more repetitive, menial Security Operation Centre (SOC) work.

Contribution #2 - BotProbe, a novel IPFIX template for botnet communication traffic capture in cloud provider networks.

Chapter 4 addressed research objective #2 through the construction of the BotProbe templates. These templates are a clear advance in technology as demonstrated by the performance test results. Evidence is provided to suggest that the algorithm attributes captured in all previous botnet detection research can be captured more efficiently with IPFIX, whilst still maintaining the original integrity of the detection algorithms. Efficiencies come not only in reduced data volumes, but also from improvements in data capture processing times. The BotProbe template exhibited an average reduction in data volumes of 14.06% ± 0.01%, with a processing time reduction of 26.73% ± 0.03% against NetFlow v5. Against PCAP, the BotProbe template measured an average reduction in data volumes of 92.95% ± 0.22%. These empirical results were measured from publically available datasets, permitting high repeatability of these results.

A high-speed data network requires multiple traffic capture devices to be distributed across the infrastructure. Multiple probes can capture TBs of data during the course of a day, making threat detection in network traffic a big data challenge. The evidence presented in Chapter 4 would indicate that BotProbe not only reduces data storage for CSPs, but has the potential to turn big data analysis into manageable data analysis.

IPFIX has the potential to change the dynamics of botnet detection. Up until now, botnet detection algorithms have been constructed based upon the attributes available through the traffic capture techniques. Where PCAP is used, all attributes in an entire network packet are available to the detection algorithm, although at a cost of high data volumes. With NetFlow v5, data volumes captured are considerably reduced, although at the cost a fixed subset of 18 attributes available to the detection algorithm. IPFIX offers the best of both solutions, allowing more granular capture of any of the attributes in a network packet, through IEs and EEs. Thus resulting in reductions in capture data volumes. This means that detection algorithm creation is no longer limited to the attributes available in the capture mechanism. This study demonstrates how detection algorithms can now dictate which attributes to capture; opening the opportunity for creation of more accurate detection algorithms that utilise data from multiple layers of the OSI (Open Systems Interconnection) model. As new application layer EEs are created for threat detection, there is scope for IANA to consider the standardisation of some of these EEs to ensure IPFIX template portability. Versions of the BotProbe templates were presented at BotConf 2015 (*Graham, Winckles and Sanchez, 2015b*).

Contribution #3 - BotStack, a novel, modular IPFIX export framework for botnet communication traffic capture in cloud provider networks.

Chapter 5 addressed research objective #3 through the construction of the BotStack IPFIX framework. This modular framework architecture is built upon open source components commonly found in CSPs, to allow IPFIX to be built into existing cloud stacks. *Steinberger, et al.* (2013) claim that flow protocols, such as NetFlow, are used by over 80% of network operators to capture network traffic management and reporting statistics. This familiarity makes the migration shift from NetFlow to IPFIX a small step for CSPs, rather than a huge uplift of the entire network to a new unfamiliar technology.

The concept of constructing a traffic capture mechanism for CSPs came from conversations with several cloud providers, who articulated that a current challenge is the detection of botnets in a multi-tenant environment with data privacy sensitivities. More recently, cloud providers have been approach by customers offering some degree of access to payload data in return for enhanced security protection; tempered by the proviso that payload access is for legitimate detection purposes and this information does not leak out of the trusted CSP domain or erode civil liberties. In

September 2016, the Anti-botnet Working Group of the CSA, undertook a survey to understand the views of over 300+ cloud customers on such trade-offs. At the time of the completion of this thesis, the results of this survey have yet to be published. Whilst the BotProbe template was constructed for use in privacy sensitive environments, the extended BotProbe template demonstrates the potential of IPFIX to capture detailed application layer information on botnets.

A comprehensive security strategy is built upon defence in depth, with multiple security solutions working together. BotStack is built to complement other security solutions that CSPs may choose to deploy such as AV, IPS or IDS. The BotStack framework was presented at IEEE INDIN 2015 (*Graham, Winckles and Sanchez, 2015a*).

Contribution #4 - Empirical evidence for siting IPFIX exporters on the host device hypervisor for maximum traffic visibility.

Chapter 5 also addressed research objective #4. Key traffic visibility profiles in a network were assigned a weighted factor. With IPFIX exporters at various locations across a network infrastructure, empirical data was gathered to understand the capture probes visibility of network traffic. Evidence gathered indicates that maximum traffic visibility is obtained with an IPFIX exporter on each network device. However, the *optimum* siting of the exporters in a multi-tenant CSP network, for maximum traffic visibility for the least number of exporters, comes from siting IPFIX exporters on the host device hypervisor connected to a vswitch tap port. In a large network, an increase in distributed exporters not only means higher data volumes captured, but data requires more co-ordinated prior to analysis as probe numbers increase. Traffic capture through fewer exporters further addresses the big data challenge in traffic capture. Placing an exporter within a tenant VM not only raises both privacy and surveillance concerns, but makes the exporter highly visible to attack. A distinct advantage of siting an exporter outside the virtualised environment is it lowers this attack visibility. Additionally, siting an exporter on a CSP infrastructure host device further reduces a probes visible attack surface to network-based malware. Exporter location optimisation was performed on a flat network, with no broadcast domain restrictions or VLANs, which may impact traffic visibility.

Chapter 5 also demonstrated that exporters sited on a host device are less susceptible to timer tampering attacks. Malware that specifically attacks flow export devices is as of yet unknown. However, as the concept of network as a sensor gains

traction, attacks against flow export should be anticipated. Although contribution #1 highlighted the security features that IPFIX offers over NetFlow, work is required to understand possible flow export attack vectors, particularly on public infrastructures such as smart cities. Work is also necessary to understand the impact upon flow traffic visibility should one of a distributed set of exporters be compromised, possibly suggesting a device to monitor all exporters against attack. Exporter siting location findings were presented at IEEE INDIN 2015 (*Graham, Winckles and Sanchez, 2015a*).

7.2 Limitations of the Study

The statistical analysis results used towards the BotProbe template creation were reliant upon the datasets that are analysed. This study chose to use datasets provided by CTU University, Prague, for reasons outlined in chapter 4. The reliability of the datasets are not in question; the source was reliable and these datasets have been used in other academic studies, including *García and Pechuocek* (2016); *Kirubavathi and Anitha* (2016); *Haddadi and Zincir-Heywood* (2015). Using a dataset that is available to other academics ensures high repeatability of the results in this study by other researchers.

One consideration of the dataset was that, although CTU University captured this data from real botnets in the wild, capture was performed in a laboratory environment. This makes the datasets cleaner than datasets captured in a live network, where the number of background processes in the laboratory network would be less than expected in a real world network. In a real world network, more background noise may be represent from applications, such as beaconing (see the following section on Future Work). Background noise was not deemed to impact the results of this study, as the high number of botnet flows analysed should minimise the influence from non-bot traffic. VirusTotal confirmed that each sample in the study does indeed contain botnet malware. Appendix B lists the suspected bot variants for the bot samples analysed during template creation. CTU continue to issue new datasets on a regular basis and work should continue to correlate these new datasets to further enhance the capture features of the BotProbe templates. The Zeus bot that formed the dataset in Chapter 6 was from the older Zeus C&C bot, as opposed to the more recent P2P GameOver Zeus bot. There are few bot C&C servers available for academic research, Zeus C&C server was one such readily available malware. Acquisition methods of more recent C&C server software is ethically questionable.

The templates constructed in chapter 4 and the framework constructed in chapter 5 were both presented with limitations by the IPFIX tools that are available through open source. The scope of the framework confined the study to open source tools to allow code modification as required. The number of open source tools that support IPFIX in any real capacity, such as a wide range of traffic contextual IEs and support for EE creation, was limited to nProbe or YAF. Commercial IPFIX tools provided no advantage in this study. YAF was chosen for this study for reasons outlined in Chapter 5. The study could have been undertaken with nProbe, although with a lower number of IEs and EEs available for study. Even with the tools available, this research only studied a proportion of the 433 IEs defined by IANA, and the almost limitless number of EEs available for construction. Overall, the number of IEs and EEs available to study was sufficient to construct an effective template. Work continues to create new EEs to extend this study.

Another limitation in the operation of BotProbe is a challenge faced by all botnet researchers; that of botnet detection evasion techniques. Signature detection and traffic capture detection are usually subverted through payload encryption. As BotProbe has been designed to capture traffic on a local area network infrastructure, the likelihood of traffic being encrypted is lower. However, payload encryption is primarily a limitation to the extended BotProbe template that uses application layer protocols from the payload, rather than just packet header information that is unlikely to be encrypted. If packet header data is encrypted, both templates will be impacted. A typical scenario could be a bot using a Virtual Private Networks (VPNs) or a Tor style network, to conceal IP addresses (*Casenove and Miraglia, 2014*). Most of the detection experiments listed in Table 2 also suffer from this limitation. Bots may employ HTTPS to evade detection, however techniques such as TLS inspection have been used to decrypt HTTPS traffic on some devices. The impact of encryption on data capture was out of scope for this work, but does require further study. A method of limiting the impact of encryption would be to limit the template to packet header data only, as in the BotProbe template, although this opens up an argument about how restricting botnet capture to layer 3 traffic attributes could impact the performance of detection algorithms. Other probe evasion techniques may exist through flow integrity tampering, such as introducing deliberate delay, insertion of fake packets or timing attacks. These should be mitigated by built-in security features of IPFIX such as SCTP, as documented in RFC-6526 (*Internet Engineering Task Force, 2012*).

7.3 Future Work

The contributions to knowledge established from this research project open up several further avenues for research.

7.3.1 CLOUD NEUTRALISATION ECO-SYSTEM

A research project is underway at Anglia Ruskin University to construct a conceptual eco-system to protect CSPs from botnets, as described in Figure 42. The research presented in this thesis forms the traffic capture element of this eco-system. An advantage of IPFIX over both NetFlow and PCAP, as described in Chapters 3 and 4, is that the highly structured format of exported IPFIX data presents itself favourably for analysis. Work is in progress to interface exported IPFIX data with deep learning neural network algorithms. There are two distinct areas where machine learning is applicable to the construction of the eco-system.

The BotProbe template constructed within this study has been constructed from a snapshot of previous botnets. But, bots are adaptive adversaries. As bot technology evolves to evade detection techniques, new attributes will become available that indicate new characteristics of these botnets. Work is underway to understand how machine learning can be applied to adaptive capture templates. A machine learning algorithm will monitor both traffic field occupancy and field variable duplicity correlation to adapt the capture template in real-time to ensure the most relevant traffic attributes are captured for feed into a botnet detection algorithm. This will involve the construction and study of new protocol specific IPFIX EEs, in particular for HTTP and SMTP.

Secondly, work is underway to apply machine learning to botnet detection. An algorithm takes threat intelligence information from honeypots, networks and other open source threat intelligence feeds to determine the most efficient and effective botnet characteristics and signature profiles. This algorithm will feedback into the adaptive capture engine to ensure optimum attributes feature in the IPFIX template, and feedforward into a neutralisation engine that uses Software Define Networking to dynamically reconfigure the network to contain the threat or forward the threat onto a honey-net for further analysis.

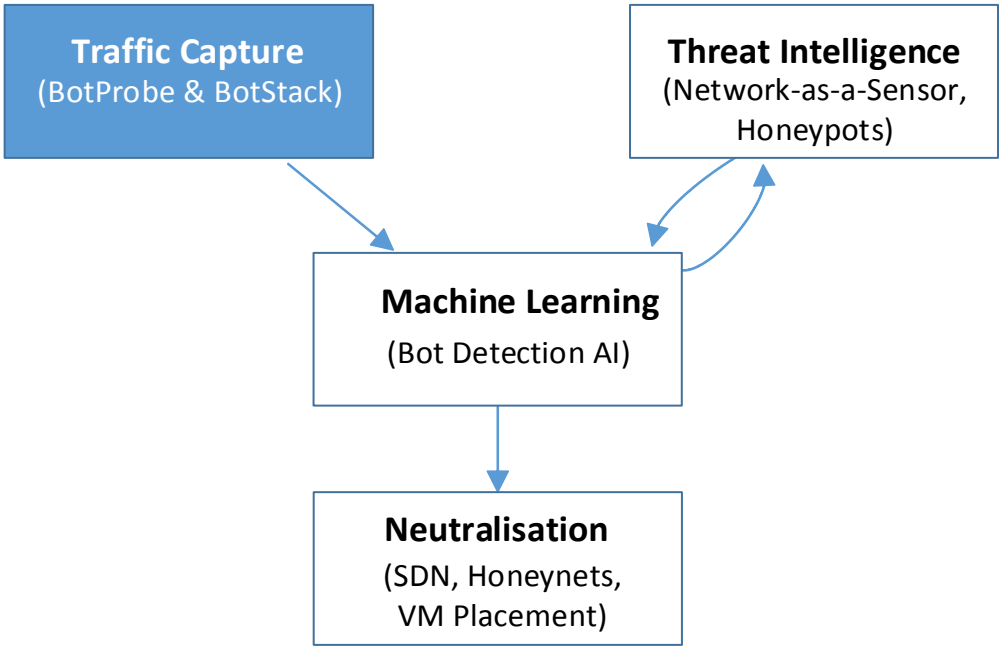


Figure 42. Conceptual botnet mitigation eco-system.

BotProbe has potential application to the field of network forensic investigations. PCAP is often used in legal interception, such as ISP lawful data interception. However, huge data volumes are a drawback as the data requires both storage and analysis. Data reduction achievable through BotProbe positions IPFIX templates as either a replacement for PCAP as a more targeted data interception approach, or as a complimentary tool in packet capture for PCAP indexing; to support the retrieval of data in PCAP data mines. Similarly, reduced data volumes from targeted data capture through BotProbe open up new applications in pre-attack forensics. The data volumes involved in PCAP prohibit the use of packet capture to capture network forensics before an attack. As BotProbe is able to reduce data volumes by the quantities shown in this study, IPFIX could be left to continually export network parameters for use in attack post-mortems to understand how the threat was able to infiltrate the network. In particular, this study demonstrated in Chapter 5 how IPFIX export can be incorporated into the hypervisor.

7.3.2 BOTNET DETECTION

The flexibility in IPFIX template construction will allow new botnet traffic characteristics to be captured that have as yet been unavailable for detailed study due to difficulty of capture in NetFlow v5. This, in itself, opens up several avenues for further work:

- Benign application beaconing, such as keep-alives, can impact botnet detection algorithms that rely on network traffic attributes. The capability of IPFIX to query traffic at the application layer could improve detection algorithms by reducing false positives associated with application beaconing;
- A botnet takedown strategy is dependent upon the botnet population size. In turn, bot population measurement depends on the accuracy of the detection technique (*Koo, Chang and Liao, 2012*). IPFIX accommodates capture of IPv6 attributes that may go to addressing some of the shortfalls in botnet population estimates that occur from IPv4 techniques such as NAT, DHCP and dynamic DNS;
- Graph databases are particularly suited to the data structures and queries in security threat analysis (*Collins, 2014*), as evidenced in Chapter 6. If IPFIX can automate the study of the less understood traffic attributes, it may lead to advances in understanding of graph theoretic structures as a method to model and detect botnets. Developing a graph theory-based algorithm for detecting bot clusters in a network represented as a digraph or adjacency table lends itself to rigorous analysis by complex queries;
- Software-based IPFIX exporters have application beyond botnet detection. A software approach to IPFIX probes make them cheaper to deploy than the "thick" probes currently used to collect PCAP. The small footprint of the core IPFIX export software means it can be deployed on low power devices, such as Internet of Things sensors. Likewise, small footprint, low CPU requirement IPFIX export software could be applied to Industrial Control System infrastructure to monitor key characteristics against attack. Work is underway to study the suitability of IPFIX EEs to capture SCADA and Modbus traffic for threat detection in vulnerable critical infrastructure systems.

7.4 Concluding Remarks

Security models change to adapt to new adversaries. Security is no longer about simply securing endpoints; the number of networked and interconnected devices is rising and technologies such as IPv6 and the IoTs means there will soon become too many end devices to reliably protect. Security is no longer about securing the network perimeter; AV software, IDS and firewalls become more difficult to implement as the demarcation of the network perimeter become blurred. Malicious attack profiles are changing. New technologies and poor security practices introduce new vulnerabilities on top of existing unpatched vulnerabilities. The Mirai botnet was one such example of exploiting poor security implementations, with the impact felt worldwide. Reliance upon CSP and IoT services is only going to increase. Signature-based anti-virus, provide some level of protection to end devices, but contributes little towards botnet eradication. In an ever changing threat landscape, new techniques are needed to complement existing security methods.

The impact from the contributions to knowledge of research project are far reaching. Botnets are a threat at an economic level for business and organisations, as well a societal level threat for an increasing amount of users that reply on services outsourced to the cloud. Internet Service Providers have a voluntary code of conduct to tackle the botnet threat. Whilst no such code exists for Cloud Service Providers, the advanced made through this research project are the cornerstone for a CSP botnet mitigation platform. The results from this study demonstrate opportunities for new and more accurate botnet detection algorithms using botnet characteristic attributes that have up to now been difficult to capture. In January 2017, the outputs from this research study were contributory towards obtaining funding from the UK government Department of Culture Media and Sport (DCMS) and Innovate UK, to benchmark the viability of commercialising this research. Security threat analysis in high-speed data networks tends towards a big data challenge. Commercial interest in overcoming this challenge was high enough to secure a second funding phase, available later in 2017, to continue the development of BotProbe.

The original contributions derived from evidence presented from this research project have established that: ***IPFIX offers clear advantages over NetFlow when applied to botnet communication traffic capture in cloud service providers networks.***

References

Altman, D.G. and Bland, J.M., 1995. Statistics Notes: The normal distribution. British Medical Journal, 310(6975), p.298.

Amazon, 2009. Zeus Botnet Controller. [online] Available at: <http://aws.amazon.com/security/security-bulletins/zeus-botnet-controller/> [Accessed 20 October 2014].

Arbor Networks Inc., 2016. Arbour Networks Releases Global DDoS Attack Data for 1H 2016. [online] Available at: <https://www.arbornetworks.com/arbor-networks-releases-global-ddos-attack-data-for-1h-2016> [Accessed 15 August 2016].

Association of National Advertisers and White Ops, 2016. The Bot Baseline: Fraud in Digital Advertising. [pdf] Available at: <https://www.ana.net/getfile/23332> [Accessed 12 June 2016].

Beloglazov, A. and Buyya, R., 2010. Energy efficient allocation of virtual machines in cloud data centers. In: *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid).* Melbourne, Australia, 17-20 May 2010. IEEE.

Bilge, L., Balzarotti, D., Robertson, W., Kirda, E. and Kruegel, C., 2012. Disclosure: Detecting botnet command and control servers through large-scale netflow analysis. In: *Proceedings of the 28th Annual Conference on Computer Security Applications.* Orlando, USA, 3-7 December 2012. New York: ACM.

Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M. and Wang, L., 2010. On the analysis of the zeus botnet crimewave toolkit. In: *Eighth Annual International Conference on Privacy Security and Trust (PST).* Ottawa, Canada, 17-19 August 2010. IEEE.

Botnet, C., 2012. *Internet Census 2012.* [online] Available at: <http://internetcensus2012.bitbucket.org/paper.html> [Accessed 23 November 2015].

Brockhus, T., 2015. Quality of measurement data on vendor-independent flow exporters.

Brook, J-M., Field, S., ShackleFord, D., Hargrave, V., Jameson, L. and Roza, M., 2016. *The Treacherous Twelve: Cloud Computing Top Threats in 2016*. [pdf] USA: Cloud Security Alliance. Available at: <https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pdf> [Accessed 13 August 2016]

Bryan. D. and Anderson M., 2010. *Cloud Computing – A Weapon of Mass Destruction*? [pdf] USA: DEFCON. Available at: <www.defcon.org/images/defcon-18/dc-18-presentations/Bryan-Anderson/DEFCON-18-Bryan-Anderson-Cloud-Computing.pdf> [Accessed 7 December 2015].

Casenove, M. and Miraglia, A., 2014. Botnet Over Tor: The illusion of hiding. In: *6th International Conference on Cyber Conflict (CyCon).* Tallinn, Estonia, 3-6 June 2014. IEEE.

Chickowski, E., 2016. *Wekby 'Pisloader'Abuses DNS.* [online]. Available at: <http://www.darkreading.com/threat-intelligence/wekby-pisloader-abuses-dns/d/d-id/1325729> [Accessed 2 December 2016].

Cid, D., 2016. Large CCTV Botnet Leveraged in DDoS Attacks. *Sucuri.net Sucuri Blog,* [blog] 27 June, Available at: <https://blog.sucuri.net/2016/06/large-cctv-botnet-leveraged-ddos-attacks.html> [Accessed 1 July 2016].

Cisco, 2015. *Cisco Network as a Sensor.* [pdf] USA: Cisco. Available at: <https://www.cisco.com/c/dam/en/us/solutions/collateral/enterprise-networks/enterprise-network-security/at-a-glance-c45-734476.pdf> [Accessed 15 December 2016].

Citrix Systems, Inc., 2015. XenServer. [computer program] XenServer. Available at: <http://xenserver.org/open-source-virtualization-download.html> [Accessed 16 May 2014].

Clegg, F., 1995. *Simple statistics: A course book for the social sciences.* 13th ed. Cambridge: Cambridge University Press.

Cloud Security Alliance, 2011. *Security Guidance for Critical Areas of Focus in Cloud Computing V3.0.* [pdf] USA: CSA. Available at: <https://downloads.cloudsecurityalliance.org/initiatives/guidance/csaguide.v3.0.pdf> [Accessed 11 November 2016].

Cloud Security Alliance, 2015. *Cloud Adoption Practices and Priorities Survey Report.* [pdf] USA: CSA. Available at: <https://downloads.cloudsecurityalliance.org/initiatives/surveys/capp/Cloud_Adoption_Practices_Priorities_Survey_Final.pdf> > [Accessed 11 November 2016].

Cohen, F., 1987. Computer Viruses: Theory and experiments. *Computers & Security*, 6(1), pp.22-35.

Cohen, J., 1988. *Statistical power analysis for the behavioural sciences.* 2nd ed. Hillsdale: Lawrence Earlbaum Associates.

Collins, M., 2014. *Network security through data analysis: Building situational awareness*. California: O'Reilly Media, Inc.

Collins, M.P. and Reiter, M.K., 2007. Hit-list worm detection and bot identification in large networks using protocol graphs. In: *International Workshop on Recent Advances in Intrusion Detection (RAID)*. Gold Coast, Australia, 5-7 September 2007. Springer.

Communications Security, Reliability and Interoperability Council, 2013. *U.S. Anti-Bot Code of Conduct (ABC) for Internet Service Providers (ISPs).* [pdf] Available at: <https://transition.fcc.gov/bureaus/pshs/advisory/csric3/CSRIC_III_WG7_Report_March_%202013.pdf> [Accessed 17 May 2015].

Constantin, L., 2014. *Over 30 Vulnerabilities Found in Google App Engine.* [online] Available at: <http://www.infoworld.com/article/2857515/cloud-computing/over-30-vulnerabilities-found-in-google-app-engine.html> [Accessed 11 November 2016].

Cooke, E., Jahanian, F. and McPherson, D., 2005. The Zombie Roundup: Understanding, detecting and disrupting botnets. In: *Steps to Reducing Unwanted Traffic on Internet (SRUTI) Workshop.* Cambridge, USA, 7 July 2005. California, USA: USENIX.

Dagon, D., Feamster, N., Lee, W., Edmonds, R., Lipton, R. and Ramachandran, A., Georgia Tech Research Corporation, 2016. *Methods and systems for detecting compromised computers*. U.S. Pat. 20,160,156,660.

Dancey, C.P. and Reidy, J., 2007. *Statistics without maths for psychology.* 4th ed. Essex: Pearson Education Ltd.

Dell SecureWorks Counter Threat Unit Research Team, 2014. *Hacker Hijacks Synology NAS Boxes for Dogecoin Mining Operation, Reaping Half Million Dollars in Two Months.* [online] Available at: <https://www.secureworks.com/blog/hacker-hijacks-synology-nas-boxes-for-dogecoin-mining-operation-reaping-half-million-dollars-in-two-months> [Accessed 12 June 2016].

Dell SecureWorks Counter Threat Unit Threat Intelligence, 2016. *Banking Botnets: The Battle Continues.* [online] Available at: <https://www.secureworks.com/research/banking-botnets-the-battle-continues> [Accessed 12 June 2016].

Deri, L., 2003. nProbe: An open source netflow probe for gigabit networks. In: *Proceedings of the TERENA Network Conference*. Zagreb, Croatia, 19-22 May 2003. Zagreb, Croatia: CARNet.

Díaz, M., Martín, C. and Rubio, B., 2016. State-of-the-art, challenges and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*, vol. 67, pp.99-117.

Dietrich, C.J., Rossow, C. and Pohlmann, N., 2013. CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis. *Computer Networks*, 57(2), pp.475-48.

Dillon, M. and Winters, T., 2014. Virtualization of home network gateways. *Computer*, 47(11), pp.62-65.

Dübendorfer, T., Wagner, A., Hossmann, T. and Plattner, B., 2005. Flow-level traffic Analysis of the Blaster and Sobig Worm outbreaks in an internet backbone. In: *International Conference on Detection of Intrusions and Malware and Vulnerability Assessment.* Vienna, Austria, 7-8 July 2005. Berlin, Germany: Springer.

ENISA, 2016. *ENISA Threat Landscape 2015.* [pdf] Greece: ENISA. Available at: <https://www.enisa.europa.eu/publications/etl2015/at_download/fullReport> [Accessed 18 December 2016].

EC Data Retention Directive 2006/24/EC of 15 March 2006 on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending Directive 2002/58/EC [2006] OJ L105/54.

Falliere, N. and Chien, E., 2009. *Zeus: king of the bots*. *Symantec Security Response.* [pdf] USA: Symantec Corporation. Available at: <http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeus_king_of_bots.pdf> [Accessed 27 February 2015]

Fang, W., Liang, X., Li, S., Chiaraviglio, L. and Xiong, N., 2013. VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1), pp.179-196.

Federal Trade Commission, 2015. *Internet of Things Privacy and Security in a Connected World.* [pdf] Available at: < https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf> [Accessed 1 March 2015].

Feily, M., Shahrestani, A. and Ramadass, S., 2009. A survey of botnet and botnet detection. In: *Third International Conference on Emerging Security Information, Systems and Technologies2009.* Athens, Greece, 18-23 June 2009. IEEE.

Field, A., 2009. *Discovering statistics using SPSS.* 3rd ed. London: SAGE Publications Ltd.

François, J., Wang, S. and Engel, T., 2011. BotTrack: Tracking botnets using NetFlow and PageRank. In: *10th International Conference on Research in Networking.* Valencia, Spain, 9-13 May 2011. Berlin, Germany: Springer.

François, J., Wang, S., Bronzi, W., State, R. and Engel, T., 2011. BotCloud: Detecting botnets using mapreduce. In*: 2011 IEEE International Workshop on Information Forensics and Security (WIFS).* 29 November-2 December, 2011. IEEE.

Fryer, H., Stalla-Bourdillon, S. and Chown, T., 2015. Malicious Web Pages: What if hosting providers could actually do something… *Computer Law and Security Review*, 31(4), pp.490-505.

García, S. and Pechoucek, M., 2016. Detecting the behavioral relationships of malware connections. In: *Proceedings of the First International Workshop on AI for Privacy and Security.* The Hague, Netherlands, 29-30 August 2016. New York: ACM.

García, S., Uhlíř, V. and Rehak, M., 2014. Identifying and modeling botnet C&C behaviors. In: *Proceedings of the First International Workshop on Agents and CyberSecurity*. Paris, France, 6 May 2014. New York: ACM.

García-Valls, M., Cucinotta, T. and Lu, C., 2014. Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture*, 60(9), pp.726-740.

Garg, S., Peddoju, S.K. and Sarje, A.K., 2016. Scalable P2P bot detection system based on network data stream. *Peer-to-Peer Networking and Applications*, 9(6), pp.1209-1225.

Gates, C., Collins, M.P., Duggan, M., Kompanek, A. and Thomas, M., 2004. More netflow tools for performance and security. In: *Proceedings of Large Installation Systems Administration (LISA) Conference*. 14 November 2004. California, USA: USENIX.

Gautheir, T.D., 2001. Detecting trends using Spearman's rank correlation coefficient. *Environmental Forensics*, *2*(4), pp.359-362.

Ghasemi, A. and Zahediasl, S., 2012. Normality Tests for Statistical Analysis: A guide for non-statisticians. *International Journal of Endocrinology and Metabolism*, 10(2), pp.486-489.

Goebel, J. and Holz, T., 2007. Rishi: Identify bot contaminated hosts by IRC nickname evaluation. *First Workshop on Hot Topics in Understanding Botnets (HotBots).* Cambridge, USA, 10 April 2007. California, USA: USENIX.

Govil, J., 2007. Examining the criminology of bot zoo. In: *6th International Conference on Information, Communications and Signal Processing.* 10-13 December 2007. IEEE.

Govindavajhala, S. and Appel, A. W., 2003. Using memory errors to attack a virtual machine. In: *Proceeding of 2003 Symposium on Security and Privacy.* 11-14 May 2003. IEEE.

Graham, M., 2014. Cloud-based detection techniques for botnets and other malware. In: *OWASP Appsec EU 2014.* Cambridge, UK, 4 July 2014. [video online] Available at: <http://www.youtube.com/watch?v=fV5kED7nryw>.

Graham, M., Winckles, A. 2014. An analysis of pre-infection detection techniques for botnets and other malware. In: *7th International Conference on Cybercrime Forensics Education and Training*. Canterbury, UK, 10-11 May 2014. CFET.

Graham, M., Winckles, A. and Moore, A., 2014. Botnet detection in virtual environments using NetFlow. In: *7th International Conference on Cybercrime Forensics Education and Training*. Canterbury, UK, 10-11 May 2014. CFET.

Graham, M., Winckles, A. and Sanchez, E., 2015a. Botnet detection within cloud server provider networks using flow protocols. In: *IEEE 13th International Conference on Industrial Informatics*. Cambridge, UK, 22-24 July 2015. IEEE.

Graham, M., Winckles, A. and Sanchez, E., 2015b. Practical experiences of building an IPFIX based open source botnet detector. *The Journal on Cybercrime & Digital Investigations*. 1(1), pp.21-28.

Graham, M., Winckles, A. and Sanchez, E., 2017. An IPFIX template for botnet detection. (Submitted to *IEEE Transactions on Information Security and Forensics*.)

Greenhalgh, T., 1997. How to read a paper. Statistics for the Non-Statistician. II:"Significant" relations and their pitfalls. *British Medical Journal, 315*(7105), p.422.

Gu. G., Perdisci, R., Zhang, J and Lee, W., 2008. BotMiner: Clustering analysis of network traffic for protocol and structure independent botnet detection. In: *Proceedings of the 17th USENIX Security Symposium.* San Jose, USA, 28 July-1 August 2008. California, USA: USENIX.

Gu, G., Porras, P. Yegneswaran, V., Fong, M. and Lee, W., 2007. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In: *Proceedings of the 16th USENIX Security Symposium*. Boston, USA, 6-10 August 2007. California, USA: USENIX.

Gu, G., Zhang, J. and Lee, W., 2008. BotSniffer: Detecting botnet command and control channels in network traffic. In: *Proceedings of the 15th Annual Network and Distributed Systems Security Symposium.* San Diego, USA, 8-11 February 2008. California, USA: USENIX.

Guo, C., Lu, G., Wang, H.J., Yang, S., Kong, C., Sun, P., Wu, W. and Zhang, Y., 2010. Secondnet: A data center network virtualization architecture with bandwidth guarantees. In: *Proceedings of the 6th International Conference on Networking Experiments and Technologies (CoNEXT).* Philadelphia, USA, 30 November-3 December 2010. ACM.

Haddadi, F. and Zincir-Heywood, A.N., 2015. A closer look at the http and P2P based botnets from a detector's perspective. *International Symposium on Foundations and Practice of Security,* vol. 9482, pp.212-228.

Haddadi, F., Morgan, J., Gomes Filho, E. and Zincir-Heywood, A.N., 2014. Botnet behaviour analysis using ip flows: with http filters using classifiers. In: *IEEE 28th International Conference on Advanced Information Networking and Applications Workshops.* Victoria, Canada, 13-16 May 2014. IEEE.

Hang, H., Wei, X., Faloutsos, M. and Eliassi-Rad, T., 2013. Entelecheia: Detecting P2P botnets in their waiting stage. In: *Proceedings of the International Federation for Information Processing (IFIP) Networking Conference*. New York, USA, 22-24 May 2013. IEEE.

Hawking, S., 1994. *Life in the universe*. [online] Available at: <http://www.hawking.org.uk/life-in-the-universe.html> [Accessed 20 October 2014]

Hayashi K., 2013. Linux Worm Targeting Hidden Devices. *Symantec Official Blog*, [blog] 27 November, Available at: <http://www.symantec.com/connect/blogs/linux-worm-targeting-hidden-devices> [Accessed 11 November 2016].

Heimer, J-L., 2014. *The SERT Q2 Quarterly Threat Intelligence Report*. [online] Available at: <http://www.solutionary.com/resource-center/blog/2014/07/sert-q2-quarterly-threat-intelligence-report/> [Accessed 29 November 2014].

Hofstede, R., Celeda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A. and Pras, A., 2014. Flow Monitoring Explained: From packet capture to data analysis with NetFlow and IPFIX. *IEEE Communications Surveys and Tutorials*, 16(4), pp.2037-2064.

Holz, T., Steiner, M., Dahl, F., Biersack, E. and Freiling, F.C., 2008. Measurements and Mitigation of Peer-to-Peer-based Botnets: A case study on Storm Worm. In: *First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET).* San Francisco, USA, 15 April 2008. California, USA: USENIX.

Husák, M., Velan, P. and Vykopal, J., 2015. Security monitoring of http traffic using extended flows. In: *10th International Conference on Availability, Reliability and Security*. Toulouse, France, 24-28 August 2015. IEEE.

IANA, 2016. *IP Flow Information Export (IPFIX) Entities.* [online] Available at: <https://www.iana.org/assignments/ipfix/ipfix.xhtml> [Accessed 15 July 2015].

IBM, 2017. 2017 Cost of Data Breach Study: Global Overview. [pdf] Available at: <http://info.resilientsystems.com/hubfs/IBM_Resilient_Branded_Content/White_Papers/2017_Global_CODB_Report_Final.pdf> [Accessed 12 June 2017].

Iliofotou, M., Kim, H.C., Faloutsos, M., Mitzenmacher, M., Pappu, P. and Varghese, G., 2011. Graption: A graph-based P2P traffic classification framework for the internet backbone. *Computer Networks*, *55*(8), pp.1909-1920.

Imperva, 2015. *2015 Bot Traffic Report*. [online] Available at: <https://www.incapsula.com/blog/bot-traffic-report-2015.html> [Accessed 7 December 2016].

Inacio, C.M. and Trammell, B., 2010. YAF: Yet another flowmeter. In: *Proceedings of the 24th Large Installation System Administration (LISA) Conference*. Berkeley, USA, 7 Nov 2010. California, USA: USENIX.

Incapsula, 2016. *2015-16 Annual DDoS Threat Landscape Report.* [online] Available at: <https://www.incapsula.com/blog/2015-16-ddos-threat-landscape-report.html> [Accessed 18 December 2016].

Inci, M.S., Gülmezoglu, B., Irazoqui, G., Eisenbarth, T. and Sunar, B., 2015. Seriously, get off my cloud! Cross-VM RSA Key Recovery in a Public Cloud. *Cryptology ePrint Archive,* Report 2015/898, 2015.

Internet Engineering Task Force, 1991. *RFC-1272 Internet Accounting: background*. IETF [online] Available at: <https://tools.ietf.org/html/rfc1272> [Accessed 7 August 2014].

Internet Engineering Task Force, 1999. *RFC-2721 RTFM: applicability statement.* IETF [online] Available at: <https://tools.ietf.org/html/rfc2721> [Accessed 7 August 2014].

Internet Engineering Task Force, 2011. *RFC-6313 Export of Structure Data in IP Flow Information Export (IPFIX).* IETF [online] Available at: <https://tools.ietf.org/html/rfc6313> [Accessed 7 August 2014].

Internet Engineering Task Force, 2012. *RFC-6526 IP Flow Information Export (IPFIX) Per Stream Control Transmission Protocol*. IETF [online] Available at: <https://tools.ietf.org/html/rfc6526> [Accessed 16 October 2014].

Internet Engineering Task Force, 2013a. *RFC-7011 Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information*. IETF [online] Available at: <https://tools.ietf.org/html/rfc7011> [Accessed 7 August 2014].

Internet Engineering Task Force, 2013b. *RFC-7012 Information Model for IP Flow Information Export (IPFIX).* IETF [online] Available at: <https://tools.ietf.org/html/rfc7012> [Accessed 7 August 2014].

Internet Engineering Task Force, 2013c. *RFC-7013 Guidelines for Authors and Reviews of IP Flow Information Export (IPFIX) Information Elements*. IETF [online] Available at: <https://tools.ietf.org/html/rfc7013> [Accessed 7 August 2014].

Internet Engineering Task Force, 2013d. *RFC-7014 Flow Selection Techniques*. IETF [online] Available at: <https://tools.ietf.org/html/rfc7014> [Accessed 7 August 2014].

Internet Engineering Task Force, 2013e. *RFC-7015 Flow Aggregation for the IP Flow Information Export (IPFIX) Protocol*. IETF [online] Available at: <https://tools.ietf.org/html/rfc7015> [Accessed 7 August 2014].

Internet Research Task Force, 2015. *RFC-7426 Software-Defined Networking (SDN): Layers and Architecture Terminology*. IRTF [online] Available at: <https://tools.ietf.org/html/rfc7426> [Accessed 11 April 2016].

Irazoqui, G., Inci, M.S., Eisenbarth, T. and Sunar, B., 2014. Fine grain Cross-VM attacks on Xen and VMware. In: *IEEE Fourth International Conference on Big Data and Cloud Computing.* Sydney, Australia, 3-4 December 2014. IEEE.

Jasti, A., Shah, P., Nagaraj, R. and Pendse, R., 2010. Security in multi-tenancy cloud. In: *IEEE International Carnahan Conference on Security Technology.* California, USA, 5-8 October 2010. IEEE.

John, J.P., Moshchuk, A., Gribble, S.D. and Krishnamurthy, A., 2009. Studying spamming botnets Using Botlab. In: *6th USENIX Symposium on Networked Systems Design and Implementation*. Boston, USA, 22-24 April 2009. California, USA: USENIX.

Johnston, R., Kim, S.I., Coe, D., Etzkorn, L., Kulick, J. and Milenkovic, A., 2016. Xen network flow analysis for intrusion detection. In: *Proceedings of the 11th Annual Cyber and Information Security Research Conference*. Oak Ridge, USA, 5-7 April 2016. New York: ACM.

Karagiannis, T., Papagiannaki, K. and Faloutsos, M., 2005. BLINC: Multilevel traffic classification in the dark. In: *Proceedings of SIGCOMM '05 Applications, Technologies, Architectures and Protocols for Computer Communications*. Philadelphia, USA, 22-26 August 2005. New York: ACM.

Karasaridis, A., Rexroad, B. and Hoeflin, D.A., 2007. Wide-scale botnet detection and characterization. In: *First Workshop on Hot Topics in Understanding Botnets (HotBots).* Cambridge, USA, 10 April 2007. California, USA: USENIX.

Kaspersky Lab, 2016. *Kaspersky DDoS Intelligence Report for Q1 2016*. [online] Available at: <https://securelist.com/analysis/quarterly-malware-reports/74550/kaspersky-ddos-intelligence-report-for-q1-2016/> [Accessed 15 May 2016].

Katsuki, T, 2012. *Crisis: The advanced malware.* [pdf] USA: Symantec Corporation. Available at: <https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/crisis_the_advanced_malware.pdf> [Accessed 12 June 2016].

Keahey, K., 2009. Nimbus: Open source Infrastructure as a Service cloud computing software. In: *Workshop on Adapting Applications and Computing Services to Multi-core and Virtualization*. CERN, Switzerland, June 2009.

Kerr, D.R. and Bruins, B.L., Cisco Systems, Inc., 2001. *Network flow switching and flow data export.* U.S. Pat. 6,243,667.

Kirubavathi, G. and Anitha, R., 2016. Botnet detection via mining of traffic flow characteristics. *Computers & Electrical Engineering*, vol. 50, pp.91-101.

Koo, T.M., Chang, H.C. and Liao, W.C., 2012. Estimating the size of P2P botnets. *International Journal of Advancements in Computing Technology*, 4(12), pp.386-295.

Kortchinsky, K., 2015. *Escaping VMware Workstation through COM1*. [pdf] USA: Google. Available at: <https://www.exploit-db.com/docs/37276.pdf> [Accessed 17 August 2016].

Kortchinsky, K., 2009. *Cloudburst: A VMware Guest to Host Escape Story*. [pdf] USA: Black Hat. Available at: <http://www.blackhat.com/presentations/bh-usa-09/KORTCHINSKY/BHUSA09-Kortchinsky-Cloudburst-PAPER.pdf> [Accessed 27 February 2015].

Kotrlik, J.W. and Williams, H.A., 2003. The incorporation of effect size in information technology, learning and performance research. *Information Technology, Learning and Performance Journal*, 21(1), pp.1-7.

Krebs, B., 2015. *Lizard Stresser Runs on Hacked Home Routers*. [online] Available at: <http://krebsonsecurity.com/2015/01/lizard-stresser-runs-on-hacked-home-routers/> [Accessed 11 November 2016].

Kreibich, C., Warfield, A., Crowcroft, J., Hand, S. and Pratt, I., 2005. Using packet symmetry to curtail malicious traffic. In: *Fourth Workshop on Hot Topics in Networks*. Maryland, USA, 14-15 November 2005. ACM SIGCOMM.

Kumar, R., Jain, K., Maharwal, H., Jain, N. and Dadhich, A., 2014. Apache CloudStack: Open source infrastructure as a service cloud computing platform. *International Journal of Advancement in Engineering technology, Management and Applied Science*, 1(2), pp.111-116.

Kunz, J., Becker, C., Jamshidy, M., Kasera, S., Ricci, R. and Van der Merwe, J., 2016. OpenEdge: A dynamic and secure open service edge network. In: *Network Operations and Management Symposium*. Istanbul, Turkey, 25-29 April 2016. IEEE.

Kushner, D., 2013. The real story of Stuxnet. *IEEE Spectrum*, 50(3), pp.48-53.

Lee, D. and Brownlee, N., 2007. Passive measurement of one-way and two-way flow lifetimes. *ACM SIGCOMM Computer Communication Review*, 37(3), pp.17-28.

Lee, Y., Shin, S., Choi, S. and Son, H.G., 2007. IPv6 anomaly traffic monitoring with IPFIX. In: *Second International Conference on Internet Monitoring and Protection*. San Jose, USA, 1-5 July 2007. IEEE Computer Society.

Lenk, A., Klems, M., Nimis, J., Tai, S. and Sandholm, T., 2009. What's Inside the Cloud? An architectural map of the cloud landscape. In: *Proceedings of the Workshop on Software Engineering Challenges of Cloud Computing*. Vancouver, Canada, 16-24 May 2009. IEEE Computer Society.

Level 3, 2015. *Safeguarding the Internet: botnet research report 2015* [pdf] USA: Level 3 Communications. Available at: <http://www.level3.com/~/media/files/white-paper/en_secur_wp_botnetresearchreport.pdf> [Accessed 18 December 2016].

Lin, S.C., Chen, P.S. and Chang, C.C., 2014. A novel method of mining network flow to detect P2P botnets. *Peer-to-Peer Networking and Applications, Springer*, *7*(4), pp.645-654.

Linux Foundation, 2013. Xen Project 4.4.0. [computer program] Xen Project. Available at: <https://xenproject.org/downloads/xen-archives/xen-44-series/xen-440.html> [Accessed 15 May 2014].

Liu, J., Xiao, Y., Ghaboosi, K., Deng, H. and Zhang, J., 2009. Botnet: Classification, attacks, detection, tracing and preventive measures. *Journal on Wireless Communications and Networking - Special Issue, IEEE Computer Society*. 2009 Issue, pp.1184-1187.

Mai, J., Chuah, C.N., Sridharan, A., Ye, T. and Zang, H., 2006. Is sampled data sufficient for anomaly detection? In: *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*. Rio, Brazil, 25-27 October 2006. New York: ACM.

Mansfield-Devine, S., 2016. DDoS Goes Mainstream: How headline-grabbing attacks could make this threat an organisation's biggest nightmare. *Network Security*, vol. 11, pp.7-13.

Mell, P. and Grance, T., 2011. The NIST definition of cloud computing. *NIST Special Publication 800-145*.

Memarian, M.R., Conti, M. and Leppänen, V., 2015. EyeCloud: A botcloud detection system. In: *13th IEEE International Symposium on Parallel and Distributed Processing with Applications.* Helsinki, Finland, 20-22 August 2015. IEEE Computer Society.

Metz, C., 2009. *BitBucket's Amazon DDoS - What went wrong.* [online] Available at: <http://www.theregister.co.uk/2009/10/09/amazon_cloud_bitbucket_ddos_aftermath/> [Accessed 2 December 2014].

Minarik, P., Vykopal, J. and Krmicek, V., 2009. Improving host profiling with bidirectional flows. In: *International Conference on Computational Science and Engineering.* Vancouver, Canada, 29-31 August 2009. IEEE Computer Society.

Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A. and Rajarajan, M., 2013. A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, *36*(1), pp.42-57.

Nagaraja, S., Mittal, P., Hong, C.Y., Caesar, M. and Borisov, N., 2010. BotGrep: Finding P2P bots with structured graph analysis. In*: 19th USENIX Security Symposium*. Washington, USA, 11-13 August 2011. California, USA: USENIX.

Nagaraja, S., 2014. Botyacc: Unified P2P botnet detection using behavioural analysis and graph analysis. *Computer Security, Springer,* vol. 8713, pp.439-456.

Narang, P., Ray, S., Hota, C. and Venkatakrishnan, V., 2014. Peershark: Detecting peer-to-peer botnets by tracking conversations. In: *Proceedings of Security and Privacy Workshops*. San Jose, USA, 17-18 May 2014. IEEE.

Narang, P., Reddy, J.M. and Hota, C., 2013. Feature selection for detection of peer-to-peer botnet traffic. In: *Proceedings of the 6th India Computing Convention.* Tamil Nadu, India, 22-25 August 2013. New York: ACM.

Nazario, J. and Holz, T., 2008. As the Net Churns: Fast-flux botnet observations. In: *Third International Conference on Malicious and Unwanted Software*. 7-8 October 2008. IEEE.

Network Working Group, 2004. *RFC-3954 Cisco Systems NetFlow Services Export Version 9 (Informational)*. NWG [online] Available at: <https://tools.ietf.org/html/rfc3954> [Accessed 20 August 2014].

Network Working Group, 2008. *RFC-5103 Bidirectional Flow Export Using IP Flow Information Export (IPFIX).* NWG [online] Available at: <https://tools.ietf.org/html/rfc5103> [Accessed 7 August 2014].

Network Working Group, 2009a. *RFC-5476 Packet Sampling (PSAMP) Protocol Specifications*. NWG [online] Available at: <https://tools.ietf.org/html/rfc5476> [Accessed 11 April 2016].

Network Working Group, 2009b. *RFC-5610 Exporting Type Information for IP Flow Information Export (IPFIX) Information Elements.* NWG [online] Available at: <https://tools.ietf.org/html/rfc5610> [Accessed 10 November 2016].

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D., 2009. The Eucalyptus Open source Cloud-computing System. In: *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. Shanghai, China, 18-21 May 2009. IEEE Computer Society.

Osborne, C., 2015. *Adware-serving Skype Botnet Disrupted*. [online] Available at: <http://www.zdnet.com/article/adware-serving-skype-botnet-disrupted/> [Accessed 11 November 2016].

Ouedraogo, M., Mignon, S., Cholez, H., Furnell, S. and Dubois, E, 2015. Security Transparency: The next frontier for security research in the cloud. *Journal of Cloud Computing*, 4(1), pp.1-14.

Pallant, J., 2013. *SPSS survival manual.* 5th ed. Berkshire: Open University Press.

Patterson, M.A., 2012. *Unleashing the Power of NetFlow and IPFIX*. Maine: Plixer International, Inc.

Pauli, D., 2014. *Dropbox Used as Command and Control for Taiwan Time Bomb*. [online] Available at:<http://www.theregister.co.uk/2014/06/30/dropbox_used_as_command_and_control_in_taiwanese_govt_attack> [Accessed 29 November 2014].

Peat, J. and Barton, B., 2005. *Medical statistics: A guide to data analysis and critical appraisal.* 1st ed. Malden: Blackwell Publishing Ltd.

Perdisci, R., Lee, W. and Feamster, N., 2010. Behavioral clustering of http-based malware and signature generation using malicious network traces. In: *7th USENIX Symposium on Networked Systems Design and Implementation*. San Jose, USA, 28-30 April 2010. California, USA: USENIX.

Pettit, J., Pfaff, B., Wright, C. and Venugpola, M., 2015. *Accelerating Open vSwitch to Ludicrous Speed*. [online] Available at: <http://networkheresy.com/2014/11/13/accelerating-open-vswitch-to-ludicrous-speed/> [Accessed 24 August 2015].

Pfaff, B., Pettit, J., Koponen, T., Jackson, E., Zhou, A., Rajahalme, J., Gross, J., Wang, A., Stringer, J., Shelar, P. and Amidon, K., 2015. The design and implementation of Open vSwitch. In: *12th USENIX Symposium on Networked Systems Design and Implementation*. Oakland, USA, 4-6 May 2015. California, USA: USENIX.

Raff, A., 2015. New Dyre Version - Yet Another Malware Evading Sandboxes. *Seculert Blog*, [blog] 30 April, Available at: <http://www.seculert.com/blogs/new-dyre-version-yet-another-malware-evading-sandboxes> [Accessed 1 August 2016].

Ragan, R. and Salazar, O., 2014. *CloudBots: Harvesting Crypto Coins like a Botnet Farmer.* [video online] Available at: <https://www.youtube.com/watch?v=llW8rI_l0u4> [Accessed 15 August 2015].

Rajab, M., Zarfoss, J., Monrose, F. and Terzis, A. 2007. My Botnet Is Bigger than Yours (Maybe, Better than Yours): Why size estimates remain challenging. In: *First Workshop on Hot Topics in Understanding Botnets (HotBots).* Cambridge, USA, 10 April 2007. California, USA: USENIX.

Ramachandran, A., Feamster, N. and Dagon, D., 2006. Revealing botnet membership using DNSBL counter-intelligence. In: *Second Workshop on Steps to Reducing Unwanted Traffic on the Internet*. San Jose, USA, 7 July 2006. California, USA: USENIX.

Rincón, S.R., Vaton, S., Beugnard, A. and Garlatti, S., 2015. Semantics based analysis of botnet activity from heterogeneous data sources. In: *International Conference on Wireless Communications and Mobile Computing*. Dubrovnik, Croatia, 24-28 August 2015. IEEE.

Rintalan, N., 2011. *Tweaking the $!@# Out of XenServer*. [online] Available at: <https://citrix.sharefile.com/d/sbe33a16657845e4b> [Accessed 20 May 2014].

Ristenpart, T., Tromer, E., Shacham, H. and Savage, S., 2009. Hey, You, Get Off Of My Cloud: Exploring information leakage in third-party compute clouds. In: *Proceedings of the 16th ACM Conference on Computer and Communications Security.* Chicago, USA, 9-13 November 2009. New York: ACM.

Rossow, C., Dietrich, C.J., Bos, H., Cavallaro, L., Van Steen, M., Freiling, F.C. and Pohlmann, N., 2011. Sandnet: Network traffic analysis of malicious software. In: *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security*. Salzburg, Austria, 10 April 2011. New York: ACM.

Roth, T., 2011. Breaking encryptions using GPU accelerated cloud instances. In: *Black Hat Technical Security Conference.* Las Vegas, USA, 30 July-4 August 2011.

Rushby, J., 1989. Kernels for safety. *Proceedings of Symposium on Safe and Secure Computing Systems.* Glasgow, UK, October 1989. Blackwell Scientific Publications.

Rutkowska, J., 2004. Red Pill... or how to detect VMM using (almost) one CPU instruction. [online] Available at: <http://www.ouah.org/Red_%20Pill.html> [Accessed 2 December 2014].

Sailer, R., Jaeger, T., Valdez, E., Caceres, R., Perez, R., Berger, S., Griffin, J. L. and van Doorn, L., 2005. Building a MAC-based security architecture for the Xen open source hypervisor. In: *21st Annual Computer Security Applications Conference.* Tucson, USA, 5-9 December 2005. IEEE Computer Society.

Sangroudi, A.A. and Mirabedini, S. J., 2015. Botnets detection for keeping the security of computer systems based on fuzzy clustering. *Indian Journal of Science and Technology*, *8*(28), p.1.

Santos, O., 2016. *Network Security with NetFlow and IPFIX: Big Data Analytics for Information Security*. Indiana: Cisco Press.

Sefraoui, O., Aissaoui, M. and Eleuldj, M., 2012. OpenStack: Toward an open source solution for cloud computing. *International Journal of Computer Applications*, 55(3), pp.38-42.

Shiravi, A., Shiravi, H., Tavallaee, M. and Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3), pp.357-374.

Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N. and Lo Iacono, L., 2011. All Your Clouds Are Belong To Us: Security analysis of cloud management interfaces. In: *Proceedings of the Third ACM Workshop on Cloud Computing Security Workshop*. Chicago, USA, 12-21 October 2011. New York: ACM.

Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., Stiller, B., 2010. An overview of ip flow-based intrusion detection. *IEEE Communications Surveys and Tutorials,* 12(3), pp.343-356.

Steinberger, J., Schehlmann, L., Abt, S. and Baier H., 2013. Anomaly Detection and Mitigation at Internet Scale: A survey. In: *International Conference on Autonomous Infrastructure, Management and Security, Springer*, vol. 7943, pp.49-60.

Strayer, W.T., Lapsely, D., Walsh, R. and Livadas, C., 2008. Botnet detection based on network behavior. In: *Botnet Detection, Springer*, vol. 36, pp.1-24.

Symantec, 2016a. *Internet Security Threat Report, Volume 21, April 2016*. [pdf] USA: Symantec Corporation. Available at: <https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf> [Accessed 12 June 2016].

Symantec, 2016b. *Linux.Routem*. [online] Available at: <https://www.symantec.com/security_response/writeup.jsp?docid=2016-033103-0851-99> [Accessed 17 December 2016].

Symantec, 2016c. *Linux.Luabot*. [online] Available at: <https://www.symantec.com/security_response/writeup.jsp?docid=2016-090915-3236-99> [Accessed 17 December 2016].

Thomas, P., 2014. Despite the news, Your Refrigerator is Not Yet Sending Spam. *Symantec Official Blog*, [blog] 23 January, Available at: <https://www.symantec.com/connect/blogs/despite-news-your-refrigerator-not-yet-sending-spam> [Accessed 11 November 2016].

Todorov, D. and Ozkan, Y., 2013. *AWS Security Best Practices*. [pdf] USA: Amazon. Available at: <https://s3.amazonaws.com/awsmedia/AWS_Security_Best_Practices.pdf> [Accessed 12 August 2016].

Trammell, B. and Boschi, E., 2011. An introduction to IP Flow Information Export (IPFIX). *IEEE Communications Magazine*, *49*(4), pp.89-95.

Trammell, B., Boschi, E., Mark, L. and Zseby, T., 2007. Requirements for a standardized flow storage solution. In: *International Symposium on Applications and the Internet Workshops*. Hiroshima, Japan, 5-19 January 2007. IEEE Computer Society.

Trend Micro, 2016. *Global Botnet Threat Activity Map*. [online] Available at: <https://www.trendmicro.com/us/security-intelligence/current-threat-activity/global-botnet-map/> [Accessed 1 November 2016].

Trend Micro, 2014. *Dropbox Used in Delivering Upatre Malware*. [online] Available at: <http://www.trendmicro.com/vinfo/us/threat-encyclopedia/spam/566/dropbox-used-in-delivering-upatre-malware> [Accessed 29 November 2014].

Turner, V., 2015. *IDC Analysis Proof Points for the Internet of Things, October 2015.* [pdf] USA: Oracle. Available at: < https://www.oracle.com/us/assets/idc-analysis-2774134.pdf > [Accessed 18 June 2016].

Van Eeten, M., Bauer, J.M., Asghari, H., Tabatabaie, S. and Rand, D., 2010. The role of internet service providers in botnet mitigation an empirical analysis based on spam data. *TPRC, 2010.*

Vaquero, L.M., Rodero-Merino, L. and Morán, D., 2011. Locking the Sky: A survey on IaaS cloud security. *Computing*, *91*(1), pp.93-118.

Velan, P., 2013. Practical experience with IPFIX flow collectors. In: *Proceedings of the 2013 IFIP/IEEE International Symposium on Integrated Network Management*. Ghent, Belgium, 27-31 May 2013. IEEE.

Velan, P., Jirsík, T. and Čeleda, P., 2013. Design and evaluation of http protocol parsers for IPFIX measurement. *Advances in Communication Networking, Springer*, vol. 8115, pp.136-147.

Verizon, 2016. *Verizon's 2016. Data Breach Investigations Report.* [pdf] USA: Verizon. Available at: <http://www.verizonenterprise.com/resources/reports/rp_DBIR_2016_Report_en_xg.pdf> [Accessed 1 November 2016].

Von Neumann, J. and Burks, A.W., 1966. Theory of self-reproducing automata. *IEEE Transactions on Neural Networks*, *5*(1), pp.3-14.

Wang, L., Tao, J., Kunze, M., Rattu, D. and Castellanos, A.C., 2008. The Cumulus Project: Build a scientific cloud for a data center. In: *First Workshop on Cloud Computing and its Applications*. Chicago, USA, 22 October 2008.

Wang, P., Sparks, S. and Zou, C.C., 2010. An advanced hybrid peer-to-peer botnet. *IEEE Transactions on Dependable and Secure Computing*, 7(2), p.113.

Wang, Z. and Lee, R., 2006. Covert and side channels due to processor architecture. *Annual Computer Security Applications Conference*, vol. 6, pp.473–482.

Wijesinghe, U., Tupakula, U. and Varadharajan, V., 2015. An enhanced model for network flow based botnet detection. In: *Proceedings of the 38th Australasian Computer Science Conference.* Sydney, Australia, 27-30 January 2015.

Wojtczuk, R. and Kallenberg, C., 2014. Attacking UEFI boot script. In: *31st Chaos Communication Congress.* Hamburg, Germany, 27-30 December 2014.

Wurzinger, P., Bilge, L., Holz, T., Goebel, J., Kruegel, C. and Kirda, E., 2009. Automatically generating models for botnet detection. *European Symposium on Research in Computer Security, Springer,* vol. 5789, pp.232-249.

Yen, T.F. and Reiter, M.K., 2010. Are Your Hosts Trading Or Plotting? Telling P2P file-sharing and bots apart. In: *30th International Conference on Distributed Computing Systems.* Genova, Italy, 21-25 June 2010. IEEE.

Zeidanloo, H.R. and Manaf, A.A., 2009. Botnet command and control mechanisms. In: *Second International Conference on Computer and Electrical Engineering, 2009.* Dubai, UAE, 28-30 December 2009. IEEE.

Zhang, J., Luo, X., Perdisci, R., Gu, G., Lee, W. and Feamster, N., 2011. Boosting the scalability of botnet detection using adaptive traffic sampling. In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security.* Hong Kong, China, 22-24 March 2011. New York: ACM.

Zhang, J., Perdisci, R., Lee, W., Luo, X. and Sarfraz, U., 2014. Building a scalable system for stealthy P2P-botnet detection. *IEEE Transactions on Information Forensics and Security*, *9*(1), pp.27-38.

Zhao, D., Traore, I., Sayed, B., Lu, W., Saad, S., Ghorbani, A. and Garant, D., 2013. Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, vol. 39(A), pp.2-16.

Appendix A: Field Counts for all IEs/EEs

The tables in this appendix contain the field counts (occupancy) for every IEs and EEs that contained data, in the bot samples analysed during this study.

TABLE 31. COMPREHENSIVE FIELD COUNT FOR EACH IE, ACROSS ALL BOT SAMPLES (BOT SAMPLES = 21, FLOW RECORDS = 7,363,521)

	sIP	dIP	sPort	dPort	protocol	appl	duration	sTimeMS	eTimeMS	packets	rPackets	bytes
CTU3_1	318602	318602	318602	318602	318602	238227	318602	318602	318602	318602	238481	318602
CTU8_1-win5	186958	186958	186958	186958	186958	46179	186958	186958	186958	186958	121767	186958
CTU8_1-win9	168065	168065	168065	168065	168065	35216	168065	168065	168065	168065	111786	168065
CTU10_1-win7	178564	178564	178564	178564	178564	166590	178564	178564	178564	178564	174904	178564
CTU10_1-win9	278687	278687	278687	278687	278687	260761	278687	278687	278687	278687	271426	278687
CTU10_1-win10	231420	231420	231420	231420	231420	205123	231420	231420	231420	231420	197115	231420
CTU16_1-win5	812996	812996	812996	812996	812996	152743	812996	812996	812996	812996	294542	812996
CTU16_1-win11	801931	801931	801931	801931	801931	158101	801931	801931	801931	801931	291038	801931
CTU25_1	288419	288419	288419	288419	288419	142420	288419	288419	288419	288419	216563	288419
CTU25_5	829624	829624	829624	829624	829624	794578	829624	829624	829624	829624	802675	829624
CTU110_4	284196	284196	284196	284196	284196	118944	284196	284196	284196	284196	271653	284196
CTU144_1	408482	408482	408482	408482	408482	76016	408482	408482	408482	408482	365564	408482
CTU145_1	411928	411928	411928	411928	411928	73448	411928	411928	411928	411928	368517	411928
CTU148_1	172287	172287	172287	172287	172287	94363	172287	172287	172287	172287	94331	172287
CTU149_1	235287	235287	235287	235287	235287	163442	235287	235287	235287	235287	177769	235287
CTU149_2	207959	207959	207959	207959	207959	155085	207959	207959	207959	207959	166476	207959
CTU160_1	206008	206008	206008	206008	206008	205741	206008	206008	206008	206008	184996	206008
CTU165_1	230475	230475	230475	230475	230475	218409	230475	230475	230475	230475	215341	230475
CTU166_1	583368	583368	583368	583368	583368	582671	583368	583368	583368	583368	561851	583368
CTU167_1	197941	197941	197941	197941	197941	188384	197941	197941	197941	197941	187061	197941
CTU168_2	330324	330324	330324	330324	330324	306592	330324	330324	330324	330324	306515	330324
TOTAL	7363521	7363521	7363521	7363521	7363521	4383033	7363521	7363521	7363521	7363521	5620371	7363521
OCCUPANCY	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**59.5%**	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**76.3%**	**100.0%**

rBytes	iFlags	rIFlags	uFlags	rUFlags	attribute	rAttrib	tcpSeq	rTcpSeq	reason	ToS	rToS	collector
238481	318602	318602	318602	318602	1734	511	318602	318602	318602	213	0	318602
121767	186958	186958	186958	186958	13958	22	186683	186886	186958	1303	0	186958
111786	168065	168065	168065	168065	13897	13	168065	168065	168065	1631	0	168065
174904	178564	178564	178564	178564	5445	1717	178554	178564	178564	23	0	178564
271426	278687	278687	278687	278687	9616	2317	278686	278687	278687	68	1	278687
197115	231420	231420	231420	231420	40220	2652	231420	231420	231420	114	0	231420
294542	812996	812996	812996	812996	154112	148	811158	812582	812996	61	0	812996
291038	801931	801931	801931	801931	157554	226	800073	801536	801931	76	0	801931
216563	288419	288419	288419	288419	47217	27229	287939	288289	288419	3402	0	288419
802675	829624	829624	829624	829624	8329	438	829587	829617	829624	1041	0	829624
271653	284196	284196	284196	284196	18403	306	283072	283824	284196	382	0	284196
365564	408482	408482	408482	408482	36902	26283	408482	408482	408482	4287	0	408482
368517	411928	411928	411928	411928	34774	24124	411928	411928	411928	4253	0	411928
94331	172287	172287	172287	172287	5588	119	172287	172287	172287	14	0	172287
177769	235287	235287	235287	235287	1826	2931	235287	235287	235287	36	0	235287
166476	207959	207959	207959	207959	752	2140	207959	207959	207959	26	0	207959
184996	206008	206008	206008	206008	17686	12940	206008	206008	206008	40	0	206008
215341	230475	230475	230475	230475	14557	245	230475	230475	230475	2022	0	230475
561851	583368	583368	583368	583368	75252	43190	583368	583368	583368	138	0	583368
187061	197941	197941	197941	197941	197941	197941	197941	197941	197941	37	0	197941
306515	330324	330324	330324	330324	35351	1876	330045	330056	330324	416	3	330324
5620371	7363521	7363521	7363521	7363521	891114	347368	7357619	7361863	7363521	19583	4	7363521
76.3%	**100.0%**	**100.0%**	**100.0%**	**100.0%**	**12.1%**	**4.7%**	**99.9%**	**100.0%**	**100.0%**	**0.3%**	**0.0%**	**100.0%**

TABLE 32. COMPREHENSIVE FIELD COUNT FOR EACH HTTP EE, ACROSS ALL BOT SAMPLES (BOT SAMPLES = 17, HTTP FLOW RECORDS = 7,167,557)

	http Server String	http User Agent	http GET	http Version	http Referer	http Location	http Host	http Content Length
CTU3_1	1053	14539	14539	14538	0	6064	14539	14290
CTU8_1-win5	0	22934	22934	22934	22713	22934	22717	0
CTU8_1-win9	0	16105	16105	16105	0	15934	16105	15937
CTU16_1-win5	0	144647	144434	149122	0	0	1642	146920
CTU16_1-win11	1	148002	147801	153298	0	2	1719	151383
CTU25_1	24588	61512	61483	61412	816	22062	61421	60811
CTU66_1	6830	23710	23710	23710	9094	1635	23710	15146
CTU109_1	0	29977	29977	29977	0	0	29977	29977
CTU110_4	2341	24022	17132	25938	10287	1997	25207	13996
CTU111_2	2498	19639	16410	20565	13586	4566	19053	14466
CTU119_3	0	14025	14025	14025	0	0	14025	14025
CTU127_2	0	33240	32222	25579	0	0	28401	220
CTU144_1	6	66128	66141	69121	55	12	66133	6497
CTU145_1	0	63665	63665	66574	68	8	63630	6391
CTU150_1	13726	0	36364	36364	0	3772	36364	36364
CTU167_1	0	186986	186986	186986	0	124644	186986	124645
CTU168_2	1	61125	61125	61125	0	22180	61125	61112
TOTAL	51044	930256	955053	977373	56619	225810	672754	712180
EEs containing data	**0.7%**	**13.0%**	**13.3%**	**13.6%**	**0.8%**	**3.2%**	**9.4%**	**9.9%**

http Age	http Accept	http Accept Lang	http Content Type	http Resp.	http Cookie	http Set Cookie	http Auth	http Via
697	14539	14539	14538	9069	1663	1303	0	651
0	22934	22934	22915	22915	0	22911	0	0
0	16105	16105	16095	16095	0	16092	0	0
0	142841	0	46671	145886	0	0	0	0
0	146103	0	39990	150341	0	0	0	1
222	54338	19352	61218	61338	11323	11532	0	178
3	9094	9094	21584	22893	189	467	5093	13
0	29977	29977	29977	29977	0	0	0	0
3745	8012	6794	23883	25937	8343	4075	2	184
2237	14720	13765	14793	20983	10016	5727	0	231
0	14024	0	11330	11330	0	0	0	0
0	11223	7007	395	25575	0	0	0	0
1530	103	775	66108	66125	32	13	0	1441
1523	87	770	63612	63628	26	13	0	1458
0	0	0	29417	32565	0	3772	0	0
0	186985	0	186950	186950	0	0	0	0
0	0	0	61112	55370	0	0	0	0
9957	671085	141112	710588	946977	31592	65905	5095	4157
0.1%	**9.4%**	**2.0%**	**9.9%**	**13.2%**	**0.4%**	**0.9%**	**0.1%**	**0.1%**

TABLE 33. COMPREHENSIVE FIELD COUNT FOR EACH DNS EE, ACROSS ALL BOT SAMPLES
(BOT SAMPLES = 15, DNS FLOW RECORDS = 8,655,304)

	A Record	NS Record	CNAME Record	SOA Record	MX Record	PTR Record	TXT Record	AAAA Record
CTU3_1	622486	271447	1478	22906	156	115790	7	125728
CTU10_1-win7	494263	0	24751	16	0	0	113043	6
CTU10_1-win9	654352	0	53	20	0	0	157131	42
CTU10_1-win10	294606	0	0	471	0	0	224012	731
CTU25_1	122532	0	569	138	0	0	0	32
CTU66_1	100047	151	756	1321	6	344	0	66738
CTU69_1	137732	0	76	34732	0	0	0	26
CTU140_2	165843	30	10658	63259	0	0	0	2
CTU141_2	182730	0	15152	66852	0	0	0	0
CTU148_1	188752	0	6	65769	0	0	0	4
CTU149_1	266240	146329	92	8163	42	308529	0	72552
CTU149_2	246477	116217	14	2255	24	274699	0	103795
CTU160_1	391625	0	0	573495	0	0	0	4
CTU165_1	432576	0	0	644655	0	0	0	4
CTU168_2	353180	188044	7263	92613	1062	0	0	77603
TOTAL	4653441	722218	60868	1576665	1290	699362	494193	447267
EEs containing data	**53.8%**	**8.3%**	**0.7%**	**18.2%**	**0.0%**	**8.1%**	**5.7%**	**5.2%**

TABLE 34. DETAILED FIELD COUNT FOR EACH SMTP EE, ACROSS ALL BOT SAMPLES
(BOT SAMPLES = 4, SMTP FLOW RECORDS = 877,827)

	SMTP Hello	SMTP From	SMTP To	SMTP Content Type	SMTP Subject	SMTP Response	Rcvd Date
CTU3_1	35709	35708	35702	26088	15892	35705	13515
CTU110_4	20240	20240	20227	5472	5293	20240	4587
CTU149_1	71458	35910	71454	4360	33624	71458	16587
CTU149_2	70351	30354	70349	0	29580	70351	7373
TOTAL	197758	122212	197732	35920	84389	197754	42062
EEs containing data	**22.5%**	**13.9%**	**22.5%**	**4.1%**	**9.6%**	**22.5%**	**4.8%**

TABLE 35. COMPREHENSIVE FIELD COUNT FOR EACH IRC EE, ACROSS ALL BOT SAMPLES
(BOT SAMPLES = 3, IRC FLOW RECORDS = 260)

	IRC TextMsg
CTU44_1	90
CTU45-1	63
CTU90-1	107
TOTAL	**260**
EEs containing data	**100.0%**

TABLE 36. DETAILED FIELD COUNT FOR EACH SSL EE, ACROSS ALL BOT SAMPLES
(BOT SAMPLES = 12, SSL FLOW RECORDS = 453,303)

	Common Name	Private Org.	Country Name	Locality Name	State Name	Street Address	Org.	Org. Unit
CTU8_1-win5	2070	0	2484	414	414	0	2484	414
CTU8_1-win9	1630	0	1956	326	326	0	1956	326
CTU25_1	2713	12	3238	555	547	2	3250	579
CTU25_5	810	0	972	162	162	0	972	162
CTU110_4	9711	639	10472	3119	3087	501	10606	7235
CTU111_2	5761	161	5815	1348	1312	79	6004	2416
CTU140_1	8873	109	9216	2760	2643	84	9247	4501
CTU140_2	6320	65	6687	1988	1738	66	6693	3391
CTU141_1	9981	69	10510	3532	3100	145	10510	5128
CTU141_2	11233	56	12008	3311	3132	66	11863	5890
CTU142_1	385	0	385	57	385	0	385	0
CTU153_1	164	0	164	0	0	0	164	0
TOTAL	59651	1111	63907	17572	16846	943	64134	30042
EEs containing data	**13.2%**	**0.2%**	**14.1%**	**3.9%**	**3.7%**	**0.2%**	**14.1%**	**6.6%**

Private Org.	Postal Code	SSL Client Version	SSL Server Cipher	SSL Cert Version	SSL Cert Serial	SSL NotValid Before	SSL NotValid After	SSL PublicKey Length	SSL Record Version
0	0	413	413	1241	1241	1241	1241	1241	413
0	0	325	325	977	977	977	977	977	325
2	0	542	542	1628	1628	1628	1628	1628	542
0	0	161	161	485	485	485	485	485	161
538	556	1694	1694	5498	5498	5498	5498	5428	1694
54	79	1133	1133	3270	3270	3270	3270	3054	1133
109	84	1516	1516	4575	4575	4575	4575	4342	1516
65	66	1081	1081	3307	3307	3307	3307	3097	1081
59	145	1720	1720	5245	5245	5245	5245	4841	1720
52	58	1767	1767	5844	5844	5844	5844	5345	1767
0	0	769	769	769	769	769	769	769	769
0	0	163	163	163	163	163	163	163	163
879	988	11284	11284	33002	33002	33002	33002	31370	11284
0.2%	**0.2%**	**2.5%**	**2.5%**	**7.3%**	**7.3%**	**7.3%**	**7.3%**	**6.9%**	**2.5%**

Appendix B: List of Botnet Samples Analysed During this Study

This appendix contains a list of all the PCAP botnet samples that were used in this study for the creation of either the BotProbe and/or the extended BotProbe templates. All botnet samples are from CTU University, Prague. The table details the date the sample was captured, the alleged bot (according to VirusTotal) and for which template the sample was analysed.

TABLE 37. A DETAILED LIST OF BOT SAMPLES USED IN THE CREATION OF BOTH BOTPROBE TEMPLATES

PCAP Sample Name	Creation Date	VirusTotal Bot	IE	HTTP	DNS	SSL	SMTP
CTU3_1	21/07/2013	Kelihos	Y	Y	Y		Y
CTU8_1-win5	10/09/2013	Zbot (?)	Y	Y		Y	
CTU8_1-win9	10/09/2013	Zbot (?)	Y	Y		Y	
CTU10_1-win7	11/07/2013	Unknown	Y		Y		
CTU10_1-win9	11/07/2013	Unknown	Y		Y		
CTU10_1-win10	11/07/2013	Unknown	Y		Y		
CTU16_1-win5	23/08/2013	Kelihos (Waledac)	Y	Y			
CTU16_1-win11	23/08/2013	Kelihos (Waledac)	Y	Y			
CTU25_1	09/09/2013	Zbot (Zeus)	Y	Y	Y	Y	
CTU25_5	10/02/2014	Zbot (Zeus)	Y			Y	
CTU66_1	07/04/2014	Sality		Y	Y		
CTU69_1	23/02/2014	Kazy (Caphaw)			Y		
CTU109_1	09/03/2015	Cridex		Y			
CTU110_4	09/04/2015	HTbot	Y	Y		Y	Y
CTU111_2	09/04/2015	Unknown		Y		Y	
CTU119_3	08/07/2015	Geodo		Y			
CTU127_2	08/07/2015	Kazy (Miuref)		Y			
CTU140_1	23/10/2015	Bunitu				Y	
CTU140_2	23/10/2015	Bunitu			Y	Y	
CTU141_1	28/09/2015	Bunitu				Y	
CTU141_2	23/10/2015	Bunitu			Y	Y	
CTU142_1	25/09/2015	Shifu				Y	
CTU144_1	23/09/2015	Shifu	Y	Y			
CTU145_1	23/09/2015	Fake uTorrent	Y	Y			
CTU148_1	26/09/2015	Zusy	Y		Y		
CTU149_1	05/12/2015	Kelihos	Y		Y		Y
CTU149_2	09/12/2015	Kelihos	Y		Y		Y
CTU150_1	05/12/2015	Tinba		Y			
CTU153_1	04/01/2016	Dridex				Y	
CTU160_1	29/04/2016	Tinba (Andromeda)	Y		Y		
CTU165_1	27/05/2016	Zeus (New Variant)	Y		Y		
CTU166_1	29/04/2016	Tinba (Andromeda)	Y				
CTU167_1	27/05/2016	Storm	Y				
CTU168_2	03/08/2016	Andromeda	Y	Y	Y		

Appendix C: SuperMediator Files

This appendix contains all of the SuperMediator configuration files used throughout the creation of either the BotProbe template and/or the extended BotProbe template.

Filename: **ie_tester.conf**

```
##########################################################################################################
# SuperMediator .conf file created by Mark Graham 11/11/2015
# Last update: 2/2/2016
#
# This file determines the attributes that will be collected by SuperMediator.
# This template exports ALL IANA defined IEs that SuperMediator is able to collect.
# NOTE: The template exports IEs 0-90, although IEs 81-90 are not defined
# in the SuperMediator documentation.
#
# There is 1 exporter:
# E1 - Information Elements              (outputs to: /flow_records.csv)
##########################################################################################################

# Define the IPFIX input file:
COLLECTOR FILEHANDLER
   PATH "out_file.yaf"
COLLECTOR END

# Define EXPORTER 1, which exports IE data to "flow_records.csv"
EXPORTER TEXT
   PATH "flow_records.csv"
   DELIMITER ","
   FIELDS 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,
31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,48,50,51,52,53,54,55,56,57,58,59,60,61,
62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90
EXPORTER END

# ENTERPRISE ELEMENTS are not exported in the BotProbe template
DPI_CONFIG
DPI_CONFIG END

DNS_DEDUP
DNS_DEDUP END

LOGLEVEL DEBUG

PIDFILE "/data/super_mediator.pid"
```

Filename: **ee_tester.conf**

```
##################################################################################################
# SuperMediator .conf file created by Mark Graham 11/11/2015
# Last update: 2/2/2016
#
# This file determines the attributes that will be collected by SuperMediator.
# This template exports ALL Enterprise Elements recognised by SuperMediator, with each element
# grouped into the appropriate protocol table.
#
# There are 2 exporters:
# E1 - Information Elements             (outputs to: /flow_records.csv)
# E2 - Enterprise Elements              (outputs to: /dpi/[tablename].txt)
##################################################################################################

# Define the IPFIX input file:
COLLECTOR FILEHANDLER
   PATH "out_file.yaf"
COLLECTOR END

# Define EXPORTER 1, which exports IE data to “flow_records.csv”
# In this configuration no IEs are collected (as FIELDS 0)
EXPORTER TEXT
   DELIMITER ","
   PATH "flow_records.csv"
   FIELDS 0
EXPORTER END

# Define EXPORTER 2, which exports EE data to “http.txt”, “dns.txt”, etc...
EXPORTER TEXT
   DELIMITER ","
   PATH "dpi"
   DPI_ONLY
   MULTI_FILES
EXPORTER END

# ENTERPRISE ELEMENTS to capture
DPI_CONFIG
  TABLE http [110,111,112,113,114,115,116,117,118,119,120,121,122,123,220,221,252,253,254,255,
256,257]
  TABLE http_extn [258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,
276,277,278,279,280]
  TABLE dns [1,2,5,6,12,15,16,28,33,43,46,47,48,50,51,53]
  TABLE DNS_other_1 [3,4,7,8,9,10,11,13,14,17,18,19,20,21,22,23,24,25,26,27,29]
  TABLE DNS_other_2 [30,31,32,34,35,36,37,38,39,40,41,42,44,45,49,52]
  TABLE irc [125]
  TABLE ftp [131,132,133,134,135]
  TABLE tftp [126,127]
  TABLE sip [155,156,157,158,159,160,161]
  TABLE smtp [162,163,164,165,166,167,168,169,170,222,251]
  TABLE ssl [185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,244,245,246,247,248,
249,250,288]
  TABLE ssh [171]
  TABLE a [54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79 ]
  TABLE b [80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99]
  TABLE c [100,101,102,103,104,105,106,107,108,109,124,128,129]
  TABLE d [130,136,137,138,139,140,141,142,143,144,145,146,147,148,149]
  TABLE e [150,151,152,153,154,172,173,174,175,176,177,178,179]
  TABLE f [180,181,182,183,184]
  TABLE g [200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219]
  TABLE h [223,224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239]
  TABLE i [240,241,242,243]
  TABLE j [281,282,283,284,285,286,287,289,290,291,292,293,294,295,296,297,298,299]
DPI_CONFIG END

DNS_DEDUP
DNS_DEDUP END

LOGLEVEL DEBUG

PIDFILE "/data/super_mediator.pid"
```

Filename: **botprobe.conf**

```
##################################################################################################
# SuperMediator .conf file created by Mark Graham 11/11/2015
#
# This is the BOTPROBE TEMPLATE configuration file used by SuperMediator.
# This exports 11 Information Elements:
#       0 = sIP
#       1 = dIP
#       4 = sPort
#       5 = dPort
#       6 = protocol
#       20 = sTimeMS
#       21 = eTimeMS
#       25 = packets
#       29 = iFlags
#       37 = tcpSeq
#       80 = collector
#
# There is 1 exporter:
# E1 - Information Elements              (outputs to: /flow_records.csv)
##################################################################################################

# Define the IPFIX input file:
COLLECTOR FILEHANDLER
   PATH "out_file.yaf"
COLLECTOR END

# Define EXPORTER 1: Output for the 11 x IEs
EXPORTER TEXT
   DELIMITER ","
   PATH "flow_records.csv"
   FIELDS 0,1,4,5,6,20,21,25,29,37,80
EXPORTER END

LOGLEVEL DEBUG

PIDFILE "/data/super_mediator.pid"
```

Filename: **extended.conf**

```
##################################################################################################
# SuperMediator .conf file created by Mark Graham 11/11/2015
#
# This is the EXTENDED BOTPROBE TEMPLATE configuration file used by SuperMediator.
# This exports 12 Information Elements:
#       0 = sIP
#       1 = dIP
#       4 = sPort
#       5 = dPort
#       6 = protocol
#       20 = sTimeMS
#       21 = eTimeMS
#       25 = packets
#       29 = iFlags
#       37 = tcpSeq
#       80 = collector
#       16 = flowKeyHash, which is used to cross reference EEs to IEs
#
# This also exports 7 Enterprise Elements:
#       1 = dnsARecord
#       6 = dnsSOARecord
#       41 = sslName
#       112 = httpGet
#       123 = httpResponse
#       125 = ircTextMessage
#       163 = smtpHello
#
# There are 2 exporters:
# E1 - Information Elements             (outputs to: /flow_records.csv)
# E2 - Enterprise Elements              (outputs to: /dpi/[tablename].txt)
##################################################################################################

# Define the IPFIX input file:
COLLECTOR FILEHANDLER
   PATH "out_file.yaf"
COLLECTOR END

# Define EXPORTER 1, which exports IE data to “flow_records.csv”
EXPORTER TEXT
   DELIMITER ","
   PATH "flow_records.csv"
   FIELDS 0,1,4,5,6,20,21,25,29,37,80,16
EXPORTER END

# Define EXPORTER 2, which exports EE data to “http.txt”, “dns.txt”, etc...
EXPORTER TEXT
   DELIMITER ","
   PATH "dpi"
   DPI_ONLY
   MULTI_FILES
EXPORTER END

# ENTERPRISE ELEMENTS to capture
DPI_CONFIG
  TABLE http [112, 123]
  TABLE dns [1, 6]
  TABLE irc [125]
  TABLE smtp [163]
  TABLE ssl [41]
DPI_CONFIG END

LOGLEVEL DEBUG

PIDFILE "/data/super_mediator.pid"
```

Filename: **nfv5.conf**

```
##################################################################################################
# SuperMediator .conf file created by Mark Graham 11/11/2015
#
# This file determines the attributes that will be collected by SuperMediator.
# This template reproduces NetFlow v5 in IPFIX.
# NOTE: It is not possible to truly replicate NetFlow5 in IPFIX, so this template aims to
# replicate fields of similar size to the original NetFlow v5 fields,
# rather than similar field content.
#
# There is 1 exporter:
# E1 - Information Elements              (outputs to: /flow_records.csv)
##################################################################################################

# Define the IPFIX input file:
COLLECTOR FILEHANDLER
   PATH "out_file.yaf"
COLLECTOR END

# Define EXPORTER 1, which exports IE data to "flow_records.csv"
EXPORTER TEXT
   DELIMITER ","
   PATH "flow_records.csv"
   FIELDS 0,1,13,15,7,37,38,52,53,4,5,29,30,6,75,33,34,31,32,81
EXPORTER END

LOGLEVEL DEBUG

PIDFILE "/data/super_mediator.pid"
```

Appendix D: Python Scripts

This appendix contains the python scripts used for the performance testing methods.

Filename: **timer.py**

```
#################################################################################################
# Python script for measuring the time taken to execute different SuperMediator templates
#
# Last edited by Mark Graham 07/03/2016
#
# Syntax: python timer.py template
#       e.g. python timer.py super_mediator_1.conf
#
# Where:
#  template     = SuperMediator template name for analysis
#################################################################################################

import timeit
import os
import sys

template = "super_mediator -c " + str(sys.argv[1])
start = []
end = []
total = []
overall = 0

for loop in range(0, 1):
   start.append(timeit.default_timer())
   os.system(template)
   end.append(timeit.default_timer())

for loop in range(0, 1):
   total.append(end[loop]-start[loop])
   print total[loop]
   overall = overall + (end[loop]-start[loop])

print "Quickest time:    {0:.4f}".format(min(total))
```

Filename: **cpu_load.py**

```
#################################################################################################
# Python script for measuring the CPU load generated by different SuperMediator templates
#
# Last edited by Mark Graham 07/03/2016
#
# Syntax: python cpu_load.py
#################################################################################################

import psutil
import numpy
import time

system_proc = []

print "Waiting for system to settle..."
time.sleep(10)

print "Capture started..."
for x in range(300):
   a = psutil.cpu_times_percent(interval=0.05, percpu=False)
   system_proc.append(a.system)

print "Capture ended..."
print "Max system %: ", max(system_proc)
```

Appendix E: Server Boot Script

This appendix contains a bash script that configures each server in the test network. This configures the physical NICs, the virtual VIFs and the NTP clock setting to allow server synchronisation.

Filename: **startup.sh**

```
#################################################################################################
# Bash script to set up Internet and LAN connections on Server1
# Created: Mark Graham 26/03/2015
# Last edited: Mark Graham: 27/08/2015 - Added NTP clock synchronisation
#
# xenbr0/eth0 = LAN (IP Address: 192.168.0.110)
# xenbr1/eth1 = <not used>
# xenbr2/eth2 = Internet (IP Address: DHCP)
#
# Syntax: bash start.sh
#################################################################################################

# Remove Xenbr0 from OVS and add it to brctl
Echo “TEST LAB CONFIG MANAGER”
echo "[+] Deleting xenbr0..."
ovs-vsctl del-br xenbr0
ifconfig virbr0 down
brctl delbr virbr0
brctl addbr xenbr0
brctl addif xenbr0 eth0
ifconfig xenbr0 up

# Stop Network Manager (Otherwise the Configuration Below Does Not Work.)
echo "[+] Stopping Network-Manager..."
stop network-manager

# Remove Legacy Configuration
ifconfig xenbr1 0
ifconfig xenbr0 0

# Configure Internet Connection
echo "[+] Connecting to Internet..."
ifconfig eth2 up
ifconfig eth2 0
ifconfig xenbr2 0
dhclient xenbr2
cat /etc/resolv.conf

# Configure LAN Connection
echo "[+] Connecting to the LAN..."
ifconfig eth0 192.168.0.110 netmask 255.255.255.0
ifconfig xenbr0 up
ifconfig eth0 0
ifconfig xenbr0 192.168.0.110 netmask 255.255.255.0
ovs-vsctl add-br xenbr0

# Synchronise with NTP Server
echo "[+] Synch NTP..."
service ntp restart
```