An Enhanced Model for Network Flow Based Botnet Detection

Udaya Wijesinghe, Udaya Tupakula, Vijay Varadharajan
Information and Networked Systems Security Research, Department of Computing,
Faculty of Science, Macquarie University, Sydney, Australia

udaya.wijesinghe@students.mq.edu.au
udaya.tupakula@mq.edu.au
vijay.varadharajan@mq.edu.au

Abstract

The botnet is a group of hijacked computers, which are employed under command and control mechanism administered by a botmaster. Botnet evolved from IRC based centralized botnet to employing common protocols such as HTTP with decentralized architectures and then peer-to-peer designs. As Botnets have become more sophisticated, the need for advanced techniques and research against botnets has grown. In this paper, we propose techniques to detect botnets by analysing network traffic flows. We developed templates for capturing traffic flows with more relevant attributes for botnet detection. Also we make use of the IPFIX standard for the specification of the templates. Hence our techniques can be used to detect different bot families with lesser overheads and are vendor neutral.

Keywords: botnet, network security, IPFIX, NetFlow.

1 Introduction

Malware is a tremendously serious intimidation to modern high-tech data networks. There are various types of malware. Among the malware, botnets are a clever piece of software, which carry out sophisticated synchronized activities while being resilient. The botnet is a group of hijacked computers, which are employed under command and control mechanism administered by a botmaster. Botnets are also known as a zombie army. Botnets are smart solution for cyber criminals since it makes available an effective mechanism to hide identity of the botnet herder; attempts for tracing back to origin always ends up at a hijacked device owned by innocent user that makes investigation all most vain.

The Botnets have been evolving during last few years and currently large varieties of botnet families are available. Botnet evolved from IRC based centralized botnets to employ common protocols such as HTTP with decentralized architectures and even peer-to-peer designs (Gu, Zhang & Lee 2008). Along with the advancement in protocols and architectures, botnets also used methods such as domain flux (Domain Generation Algorithms), IP flux (Single flux or Double flux) (Antonakakis et al. 2011) DNS techniques and encryption to evade detection.

As Botnets have become more sophisticated, the need for advanced techniques and research against botnets have grown. As a result of this, detecting botnet activities have become important part of modern security systems. Indeed fair amount of research has been performed during last few years in the context of botnet detection and prevention. In spite of all this effort and expense, detecting botnets remains a challenge. Furthermore, the nature of battleground is highly volatile and bot designers are developing more stealth botnet designs every day. Also proliferation of super-fast data networks, encryption technologies and anonymity techniques are making security professionals' life harder. Botnet operators are usually financially motivated and clever enough to evade detection. Therefore, most security researchers are still struggling to find better solution even though academia and industry have conducted significant amount of research in this area. Hence, detecting botnets have become very challenging and requires an enormous effort to find sophisticated monitoring and detection methods.

IP Flow data has been used to detect various malwares including botnets in a high speed, large volume data networks. Flows are defined as group of packets, which share common characteristics such as source IP, destination IP, source port, destination port and protocol type. Each packet that is routed within the network can be examined to identify unique flows based on conjoint attributes of the packets. These attributes are considered as the IP packet identity or fingerprint of the packet and determine if the packet is distinctive or tie with other packets.

All packets with the uniform source/destination IP address and ports, protocol, interface and class of service are categorized into a flow. Subsequently number of packets and number of bytes are matched. This approach of fingerprinting of a flow is scalable as a large amount of flow information is summarized into a database. This flow information is extremely valuable for analysing malicious behaviour of high speed, high traffic volume IP networks. As we are only summarizing flow details it is relatively very small compared to actual size of data streams. Further, encryption of payload is not impacted with accuracy of flow information.

Currently most of the vendors have their own formats for capturing the traffic flows. Cisco NetFlow v5 and other parallel versions of network flows are widely used. In these versions of network flow formats, attributes, which can capture from network devices, are pre-defined; hence those network flow formats are known as fixed flow data templates. Cisco first introduced capability of selecting attributes from larger set of attributes in order to define

customised data templates in NetFlow v9 known as flexible NetFlow. Recently IPFIX (IETF 2007) (Quittek et al. 2008) has been developed to make common standard for IP flows. IPFIX address transport protocol, security, IETF and vender specific information elements in addition to the principles of NetFlow v9 (Cisco Inc 2012) such as separate templates and records. One of the main concerns in IPFIX is to use congestion-aware protocol in order to transfer data. So Stream Control Transport Protocol (SCTP, RFC 2960) and Stream Control Transport Protocol – Partially Reliable (SCTP-PR, RFC 3760) have defined to export data to the collectors while facilitating conventional protocols; TCP or UDP can be used to transport data however SCTP-PR is preferred.

Our Contribution: In this paper, we propose a methodology to enhance botnet detection techniques by leveraging features of flexible network flows. Our approach consists of two main steps. Firstly, we identify suitable data set templates with more relevant attributes for botnet detection from IP flows, using IPFIX. Also we customize each template according to various botnet families and attributes to enhance the effectiveness of the system. This customization overcomes the limitation of fixed flow templates in the previous approaches. Further, we improve effectiveness of existing network flow based botnet detection by above templates while significantly reducing size of the dataset. Secondly, we use IP flow data to detect botnet behaviours in unlabelled traffic. To best of our knowledge, this is a novel approach and it enhances existing IP flow based botnet detection research.

Our paper makes following major contributions;

- We develop a general network flow based botnet detection framework that is based on flexible IP flows (IPFIX) in order to overcome limitations of current network flow based botnet detection systems.
- We create a generic IP flow template that is flexible for use with existing network flow based botnet detection systems. This results in minimising the dataset overhead for capturing the flow while keeping all the necessary attributes for detecting the bots.

Our work is organized as follows. In Section 2 we propose IPFIX standards based techniques for detecting attacks from a range of bot families. Section 3 presents the implementation and analysis of our model. We will also compare our model with some of the recently proposed techniques. Section 4 presents some of the important related work and Section 5 concludes.

2 Our Model

In this section we will present high-level overview of our architecture and then describe the architecture components in detail.

2.1 High level Overview

Figure 1 illustrates the high-level architecture of our model. Generic templates, Flow collector, Filtering Engine and Botnet detection engine are some of the important components in our model. Generic templates are used for capturing flow informatin using IPFIX. Flow collector is a centralised server that is used for storing and organising the data captured at different devices by using the generic templates. Filtering is used to reduce dataset by filtering out unwanted data that is not related to botnets. Finally botnet detection engine, correlates flow information using machine learning techniques to find the pattern and detect bots. We will discuss each of component in datail in the section 2.2

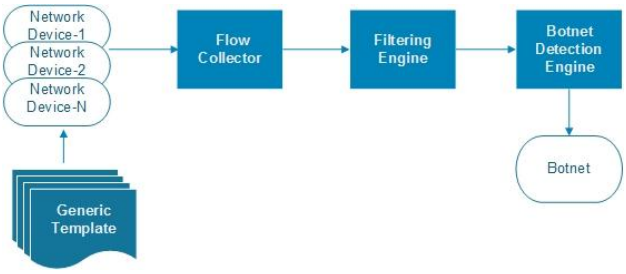

Figure 1: High-level system overview

We use offline analysis for analysing the behaviour of different bot families. We make use of virtualisation techniques to generate traffic related to the bot or use publicly available dataset related to the bot. For example, we create a test bed with several virtual machines and infect some of the virtual machines with the bot and monitor the behaviour of the infected machine. Then we use machine-learning techniques for analysing the behaviour of the bot and identify the specific features that can be used to detect the bots. We repeat the process for different bot families and identify the specific features that can be used for detecting the bots.

We have used different machine learning techniques for analysing the communication of different bot families and made some interesting observations.

As a pre-programmed system, botnet runs automatic algorithms throughout their life cycle in order to conduct different activities. Even though botnet designer makes significant effort to randomize their activities, our analysis confirms that there are still patterns to identify them.

- Some of the bot families such as IRC-based, HTTP-based and DNS-based have specific characteristics when they react to the C&C instructions and how bots react to those instructions
- Bots such as SpyEye and some versions of Zeus have a unique flow behaviour that can be used as signature for detecting the bots.
- The communication in peer-to-peer bots such as, queries for update and instruction create many small uniformed packets than other legitimate P2P communications. Also the initial exchanges of packets have a fixed format and can be easily differentiated from legitimate traffic.

After analysing a range of bots, we develop a generic template by combining all the unique features of network flows for detecting different bot families. For example, features such as source address and destination address are common for detecting any bot. Hence our generic

template includes relevant features for detecting different bot families. Also note, if we are interested in detecting specific bots, we only need to use the specific features to detect that bot. There is no need to develop a generic template for such cases. Now let us consider how the generic template can be used to detect different types of bots.

We make use of the IPFIX standard for defining the generic template. In the current state of art, different vendors have proprietary protocols such as Cisco-NetFlow (Cisco Inc 2012), Juniper-j-flow or cflowd (Juniper Networks Inc 2011), Huawei-Netstream (Huawei Technologies Co Ltd 2012) for capturing the flow traffic. Hence by defining the generic template using IPFIX, it can be applied on the devices that belong to different vendors.

The generic templates are applied to different devices such as layer 3 switches and routers. These devices analyse the traffic flowing through them and report the traffic flow information to the flow collector. Filtering is used to minimise the captured dataset by eliminating redundancies and unwanted traffic that is not related to botnets. For example, core routers can report the network flow information of communication between internal hosts and external hosts. Gateway routers can report network flow information of external hosts. Hence we only store the internal host communication information reported by the core routers and the external host communication reported by the gateway routers. Botnet Detection Engine is used for analysing the filtered traffic and detecting bots. Since generic templates are used for capturing the flow information, the captured data includes data for detecting different bots. Hence different bots can be detected by analysing subset of the features captured in the network traffic flows. The machine learning techniques that were initially used for identifying the specific features in each bot are again used for detecting the attacks from the flow dataset.

We have compared our model with some of the related techniques. There are several advantages with our approach. Our approach has minimal dataset overhead since the templates are developed from the attributes that are extracted using machine-learning techniques for efficient detection of the bots. Our model can detect a range of botnet families using various activities throughout their lifecycle including C&C interactions, recruiting new bot members and synchronized attacks. In Section 3.3, we provide a detail comparison of our model with related techniques.

2.2 Architecture Components

In this Section we will describe the components of our framework.

2.2.1 Generic templates

Templates are used for capturing network traffic flows from different devices for detecting the bots. In this paper we consider IPv4 and IPv6 traffic flows. Flows are defined as group of IP packets, which share common, attributes such as source IP, destination IP, source port, destination port, protocols type, class of service and interface of the network element. Based on these attributes each packet going through a network element can be classified into unique flows. For example, these attributes enable us to determine if the received packet belongs to an existing flow or a new flow.

Various network device vendors have developed their own IP flow capturing techniques such as, Cisco NetFlow, Juniper-j-flow or cflowd, and Huawei-Netstream. However such flow classification schemes have several limitations since they use proprietary protocols, have fixed format for flow classification and are not compatible with products from different vendors. Hence the related techniques such as Botfinder (Tegeler et al. 2012), Disclosure (Bilge et al. 2012), BotSniffer (Gu, Zhang & Lee 2008), BotTrack (Francois et al. 2011), (Zhao et al. 2012), (Zang et al. 2011), (Strayer et al. 2008), which make use of these tools for flow analysis, are not efficient.

The fixed nature of capturing IP flows limited the capability of security analysis while avoiding important features from IP flow data, For example, conventional fixed IP flows only report 20 attributes: Source IP address, Destination IP address, IP address of the next hop router, SNMP index of the input interface, SNMP index of the output interface, packet in the flow, total number of L3 bytes in the flow's packets, system up time at start of the flow, system uptime at the last packet of the flow received, layer 4 source port number, layer 4 destination port number, cumulative OR of TCP flag, IP protocol, IP type of service, AS number of the source, AS number of the destination, source address prefix mask bits, destination address prefix mask bits, and two unused attributes (pad 1 and pad 2) .

The fixed attributes consume 48 bytes for each of the flow record. From our analysis, we identified that most of the attributes that are used in the fixed flow templates for classification of the flows are not useful for detecting the bots. Furthermore, there is no way to include new attributes that can be used for efficient classification of the flows for bot detection. Hence the main disadvantages of using fixed IP flow is that it narrows down the security analysis that can be performed on the captured traffic and it also results in unnecessarily increasing the size of data set.

Recently, IETF introduced open standard for flexible IP flows named as IPFIX. This IP fixed abstracts most of the features from NetFlow version 9 to define standard for IP flows. However unlike Netflow version 9, IPFIX added variable length fields. Furthermore, IPFIX standard allow users to select wider range of attributes from more than 400 different data points (RFC7012, 2013), Hence IPFIX enables the users to define the templates with any of the attributes for traffic classification.

Our system makes use of IPFIX for developing customized templates for detecting botnets. There are two main benefits with the customized IP flow templates used in our model. First, customized templates facilitate to reduce size of the data set by removing unwanted attributes from fixed data set. This is a very important benefit of our proposed system, as this will significantly reduce the size of data sets for analysis and save space and computing power. Secondly, proposed templates

provide more room for botnet detection since new templates come up with additional important attributes in addition to traditional network flow data set. For example, customized templates allow us to develop new data templates based on botnet families, adding new features from IP header, adding new features from pay load (DNS attributes), adding new statistical attributes (average pay load size) in order to detect botnets. Further, this will facilitates to detect DGA and fast flux attacks by introducing DNS attributes in to network flow data set.

Recall that we have used machine-learning techniques for identifying the attributes that are efficient for detecting the bots. We have used these attributes for developing generic templates by using only the attributes that are efficient for detecting different bot families. Once customized templates have been developed, it is applied to different network devices. Since we make use of IPFIX for developing the template, it can be used on devices that belong to different vendors. Now the network devices classify the flows based on the attributes specified in the template and send this information to the flow collector.

2.2.2 Flow collector

The flow collector is used to collect the flow records from different devices. As flexible IP flow comes with variable size filed set and data set, flow collector enables storing according to the requirement of flow templates. The network devices send their data along with data templates to the flow collector. The flow collector software facilitates the security administrators to define storage template to match with the flow templates for storing the data. Then collector builds a raw dataset based on information from templates, by storing received data in correct order. Data is arranged into directories based on type of traffic, flow direction, year, month, date and time. Flow collector also provides additional features such as compression of the data to save the storage space and ability to perform search on the stored data. Another important feature of flow collector is to facilitate usage of various analytical tools on the stored data to eliminate unwanted traffic and detect botnets. This provides more room to use different filtering and analytical tools to access data and make data available for various analysis methods.

2.2.3 Filtering engine

The main purpose of the Filtering Engine is to reduce data set since generic templates can also capture the flows that are not related to botnet. Hence before forwarding the collected flows to the next stage for further processing and detection of attacks, we reduce the dataset by removing unnecessary data stored in the flow collector. This will make data set more manageable and save computing power in next steps while providing capability to deal with high volume data networks. Similar to (Strayer et al. 2008) we use a five step filtering mechanism for botnet detection system. However there are also some differences in the filtering used in our model. Below we summarise how our model differs to Strayer.

Firstly, Strayer mainly targets detection of IRC based botnet. Hence his technique makes use of only the TCP flows for the detection of botnet and filters out all other traffic. Our model targets a range of bot families. Hence we make use of TCP and UDP flows for the detection of bots. For example, significant amount of botnets manipulate legitimate DNS traffic to tunnel C&C communication. So filtering out of non-TCP flows is not suitable for DNS based botnet families.

Secondly, Strayer uses a filter to remove the nuisance port-scanning chaff in order to reduce dataset. The main idea behind this is if a flow contains only SYN or RST flag that means TCP flow is not established and un-established flows are no value for C&C communication. However, un-established flows represent some botnet activity such as DDOS or random packet throwing to mislead detection tools. So instead of filtering out un-established flow we categorize flows with only SYN or RST flags in order to detect botnets. Further, flexible IP flows allow us to customize flow template to collect flow data based on TCP flags while fixed flows only allow us to collect cumulative TCP flags.

Thirdly, Strayer introduced a filter to remove high volume data flows as C&C communication is not supposed to generate high bitrate data flows. Also software updates and peer-to-peer data transfer generate high volume data flows. Hence we use similar filter to distinct legitimate peer-to-peer traffic from P2P or hybrid botnets' C&C channels.

Fourthly, Strayer uses filter to remove flows with large IP packets specifically packets size above 300 bytes. This filter can only be used for detecting IRC botnets. However some botnets use large data packets to send stolen data to its C&C server. Hence we consider all the packets for our analysis. However, if specific pattern exists for some of the bots we modify this filter in order to identify such a communication by selecting correct size of the packets. Further, usage of IPFIX in our model allows us to get payload and header size of the packets separately. We use this attribute for fine-tuning of the filter.

Finally, Strayer used filter to remove flows with 2 or less packets or very brief time windows to remove port-scanning flows. In our model we make use of this filter to identify vulnerability or open port scanning by the botnets to find new victims.

Another important feature in our model is to define additional filters for C&C traffic behaviours of different botnet families. Such filtering enables to filter flows related to particular botnet family and facilitate further analysis effectively as it will significantly reduce data set.

2.2.4 Botnet Detection Engine

Once data set has been selected for comprehensive analysis by series of filters describe in previous stage, data set is fed to botnet detection engine. First step of the engine is to classify or cluster the flows in data set in order to identify similarities of the flows. The machine learning techniques, which are initially used to identify the specific features for each bot, are used in this stage to detect the specific bots. There are three main behavioural patterns; bot behaviour, botnet behaviour and temporal behaviour are used in our system.

In the bot behaviour, we analyse flows generated from one bot or single machine to identify its C&C communication or attack graphs. In botnet behaviour, we analysis flows generated by group of bots or machines, in order to detect botnet activities. Further, in above method we horizontally analyse flows generated in a network to find suspicious pattern related to botnet communication. Finally, in temporal or vertical method, we analysis flows generated by bots or botnets over the period of time in order to detect patterns. Once we identify, similarities or correlations, then we compare them with the patterns we already identified through machine learning.

We have analysed a range of bot families such as IRC, HTTP, peer-to-peer and hybrid bots and made several observations that enable for the detection of bots. For example, in case of C&C bots our analysis was mainly focused on detecting the bots during different phases in botnet life cycle (Feily, Shahrestani & Ramadass 2009); initial infection, secondary infection, connection, malicious command and control, and update and maintenance. In the initial infection phase attacker exploits vulnerabilities on victims and gains basic control over victims. Then secondary infection phase is used to further download and install malicious script and binaries to get full control of the victim. Once secondary infection is complete, bots make connection to its C&C server in order to become a member of botnet. Then bot will receive command and control from C&C server to conduct malicious coordinated activities. Finally bots update its binaries to get more functionality or evade detection. Below we summarise some of the important observations from that analysis of range of bot families:

In case of direct C&C related bots such as IRC and HTTP every bot needs to find its own C&C controller to be a member of centralized botnet. In case of peer-to-peer and Hybrid bots such as Kademila, Chord and GameOver (Zeus v3) each bots need to find its servant bot or proxy bot to get C&C instruction and become a member of P2P botnet. Our analysis confirms that these botnets are using hard coded static IP lists or/and DNS service to locate its C&C server, from which it has to receive the control commands and updates. This generates a pattern, which is used for the detection of bots. Conversely, to evade from discovery and close it down, the C&C servers' use distinctive methods like Domain flux (Domain Generation Algorithm - DGA), and IP flux to alter their DNS name or the IP addresses connected with FQDN. However, this makes the botnets to connect to different C&C servers instead of connecting to a specific one. As a result, the bots perform a large number of DNS lookups and scan a large volume of addresses to find the C&C server. These patterns are used in our model to track the botnet.

The second observation related to C&C bots is that the bots need to perform frequent communication with the C&C server. This is essential to keep control and update the botnet by the bot master. Once C&C server has been discovered, the bots take updates and commands from the botmaster about what type of action has to be executed. The action can be whatsoever from sending spam emails to intensifying a DDoS attack. Another important massage type is keep alive packets that are sent from the bot to the C&C server. Moreover, some bots (such as Zeus and SpyEye) report back to C&C server with the information they steal from compromised computers. Even though botnet designers are working hard to randomize those communications to evade detection there are some vertical or/and horizontal correlation on their communication. For example, some of the Zeus bots (Zeus v1.3) send updates at fixed interval of 20 minutes.

All the botnets desire to spread over its network and recruit more bots into its botnet, which ultimately benefits to surge the strength of a botnet and consequently that of the attack carried out too. Hence the bots scan for other machines in its network for vulnerabilities. If a vulnerable machine is found, they will run exploits to compromise the machine. When scanning the network for possible machine to infect, bots generate a burst of small packets. So this activity makes a sudden increase in the number of packets without a major increase in the traffic volume that could be used to detect bots.

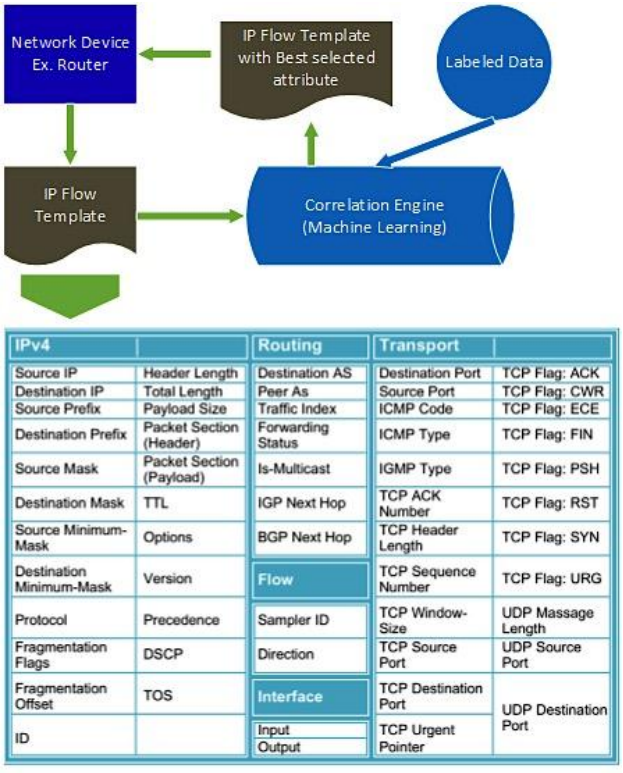

Figure 2: Theoretical framework

The bot detection engine also looks for DDoS activities such as, outbound TCP SYN packets having an invalid source IP address. The reason for these large number of TCP SYN packets could mean that some of the internal hosts in the network is part of a botnet and are participating in a DDoS attack. Some botnets such as (Grum, Bobax, Cutwail and Donbot) generate email spamming. Email spamming involves sending enormous amount of spam emails advertising fake products intended at financial gains. When the hosts in a network are part of a botnet involved with spamming, they send huge number of emails to the outside world and mostly using some external email server. So, unusual SMTP activity from the network to the outside is another significant network activity that is used for tracking the bots.

3 Implementation and Analysis

Our approach consists of two main steps. Firstly, we make an effort to identify best suitable template in order to collect flow data set. As there is no publicly available IPFIX dataset that can use for our study, we have to setup our own controlled lab environment to get a flexible IP flow dataset that includes malicious traffic sample. In such a case we obtain two separate datasets, dataset from analysis of different botnets in our lab environment and dataset from University network over a period of two days (about 5 GB), which contains everyday usage traffic patterns. We merged both the datasets using TCP replay tool to produce test dataset. We have used 10-fold cross-validation technique for training and testing of our model and achieved high detection accuracy.

We have analysed different bot families such as HTTP, IRC, and peer-to-peer bots to find best suitable template. First we have used a template with 57 attributes (see Figure 2) for capturing the traffic flows of each bot then used machine learning techniques to identify the best attributes for detecting each bot family. Theoretical framework of the system is shown in Figure 2.

We have used several machine learning techniques such as Bayesian Network, Neural Network, Support vector Machine, Gaussian and Nearest Neighbour classifier for identifying the best attributes for each bot since none of the machine learning technique was found to be effective for detecting all the bot families. For example, Decision Tree classifier is found to be effective for detecting peer-to-peer botnet by analysing the flow intervals. SVM and Bayesian networks were found to be effective for detecting C&C communication of the centralized botnets such as HTTP and IRC botnets.

3.1 Lab Environment

Figure 3 and Figure 4 illustrate the design of the lab, which has been used to analyse different botnet samples and collect IPFIX data sets. In this case, since Cisco devices were used in the lab environment, we made use of NetFlow 9 that is compatible with IPFIX standard.

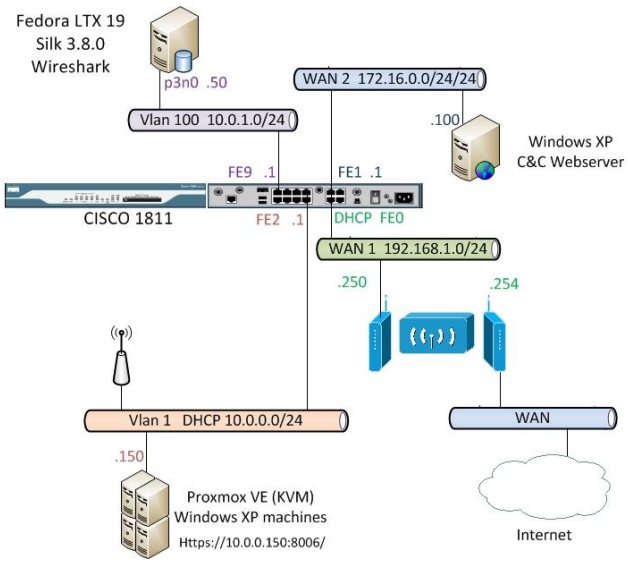

Figure 3: Lab Setup

3.1.1 Building a Flow Collector/Analyser using SiLK toolset.

We make use of SiLK for the implementation of the flow collector and used Python for developing tools to analyse the data stored in the SiLK server. SiLK is part of the Network Situational Awareness (NetSA) developed by the team of security professional at Software Engineering Institute (SEI) CERT division (NetSA 2014). The NetSA group constructs unified, standards compliant flow collection and analysis tools with open source license agreement. SiLK stands for the "System for Internet-Level Knowledge" and is cooperated of network traffic analysis tools divided into two main categories – receiving and packing flow data, and analysis. It supports efficient collection of network flow data, storage, and analysis, which lets us to very speedily query large amount of traffic data sets. We have built a SiLK v3.8.2 Collector/Analyzer on Fedora Linux v20.

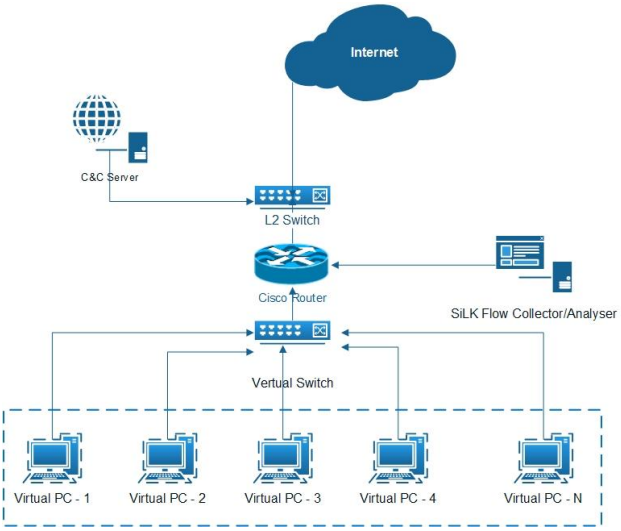

Figure 4: High-level design of the Lab

The SiLK packing system collects IPFIX data and stores it in a more space efficient "packed" format binary flat file. Mainly, we have used "rwflowpack" from SiLK which receives the flow record, converts the data to the SiLK Flow record format, categorises the flow records, and writes the records to hourly flat-files organised in a time-based directory structure. The Python tools are then used for querying and analysing the data stored in the server.

3.1.2 Building the Bots

As shown in Figure 3, we have run Windows XP virtual machines on Proxmox VE (KVM) hypervisor to build bots infected with the version of Zeus, SpyEye and Warbot. Below we present an overview of the bots.

Win32/Zeus: Zeus is also known as ZBot/WSNPoem, prominent http based attempts first discovered in 2007 (Andriesse et al. 2013). Zeus is specially designed for stealing banking information, by using web injection technology. It outbreaks by abusing vulnerabilities in the browser security to alter web pages and manipulate financial transactions by changing or adding malicious details. Form grabbing is a technique of seizing web form data in many web browsers. Later versions of Zeus were released as commercial toolkit to build and administrate a

botnet. It consists with a control panel that is used to command and control the botnet. It also has a tool to create the bot executable for infecting different machines.

Win32/SpyEye: SpyEye is http-based botnet, which appeared in 2009 (Sood, Enbody & Bansal 2013). The architecture is similar to Zeus but comes with advance features. As Zeus, SpyEye was also released as commercial crime ware kit distributed through underground web forums. It also contains a builder module for creating the bot executable with config file and a Web control panel for command and control (C&C) of a bot net. Further, it is capable of keylogging, encrypted configuration file, daily email backup and detect and kill Zeus bot files.

Win32/Warbot: Warbot (YouTube 2013) specially design for DDoS attacks, in addition it let botmaster to run other files on the victim machine so it has a Loader capabilities too. This bot is cracked and available to download online.

We have infected the lab machines with different bot executable files and also reverse engineered to analyse the bot behaviour. We have been able to determine the communication patterns of the bots with the C&C server by information captured from the compromised devices. The Builder tool is used to build the encrypted dynamic configuration file and the bot executable file. For example, the Zeus Builder first checks if the system is already infected with the Zeus and the version of the infected file. Figure 5 shows the system information reported by the builder.

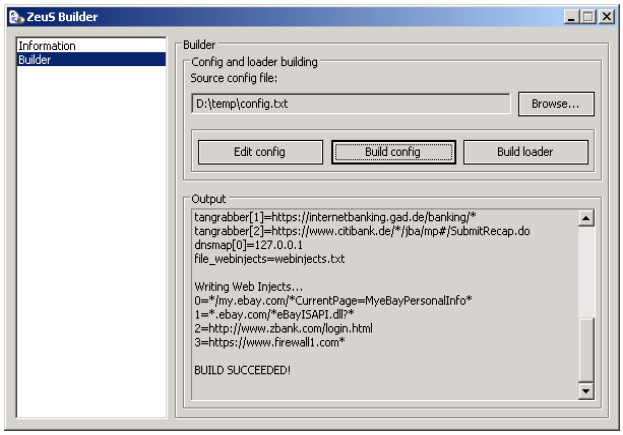

Figure 5: Zeus bot builder

The Zeus Control Panel shown in Figure 6 is mainly used to track the state of controlled botnets and to send script commands to the bots. It also provides an organized way to view and access information collected by the bots from infected computers.

3.1.3 Building the C&C Server

We have used separate Windows XP virtual machine to built C&C Server. The Control Panel is an open source PHP application that is running on an IIS or Apache web server. Some additional software and MySQL user with appropriate permissions is also used in the setup. When the system is ready, the Control Panel code was copied into the web server directory. The install page can then be accessed from a browser. Once this form is completed the remainder of the setup is done automatically. Figure 6 shows the screen shots of C&C server setup window for Zeus botnet.

The router in Figure 4 is configured to capture the flows using NetFlow v9 and send them to SiLK flow collector (rwflowpack). Then the SiLK packing system stores it in a more space efficient "packed" format binary flat file in SiLK Flow record format while categorizing the flow records, and writing the records to hourly flat-files. Filtering Engine is used to reduce the data set by filtering out the data that is not useful for detecting bots. The filtered dataset is clustered and further analysed to find correlations between flows. Finally these correlations is used to identify botnet activates and victimized hosts in the network.

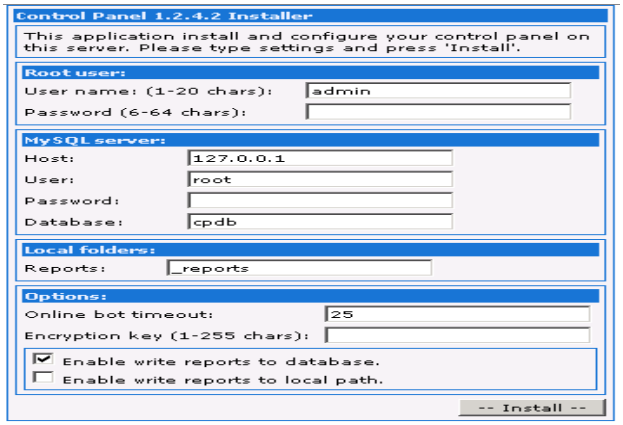

Figure 6: Zeus Control - Installation configuration

3.2 Results

Figure 7 shows the generic template that was developed to detect different bots for IPv4 protocol. We have only used the best attributes for detecting different bot families such as Zeus, SpyEye and Warbot. Figure 8 shows a sample flow record that is captured by the generic template and forwarded to the flow collector. The total length of the dataset is only 30 bytes. This is of significant advantage compared to related techniques since the size of the dataset for these cases is 48 bytes in fixed templates. Also, several attributes that are used in the fixed template are not efficient for detecting the bots. Furthermore the related techniques can only detect specific bot families. Our model can detect bots from different bot families.

```
Flow Exporter eTest2:
  Client: Flow Monitor mTest2
  Exporter Format: NetFlow Version 9
  Template ID    : 256
  Source ID      : 0
  Record Size    : 30
  Template layout
```

Field	Type	Offset	Size
ipv4 source address	8	0	4
ipv4 destination address	12	4	4
transport source-port	7	8	2
transport destination-port	11	10	2
ip protocol	4	12	1
transport tcp flags	6	13	1
counter bytes	1	14	4
counter packets	2	18	4
timestamp sys-uptime first	22	22	4
timestamp sys-uptime last	21	26	4

Figure 7: Generic template

IPFIX can also be used to capture specific bytes from the application layer. Hence the generic templates in our model can be easily extended to detect various bot families who use ICMP, SNMP, IPv6 or DNS as

communication channels to get C&C by considering the application layer attributes, IPv6 attributes, ICMP attributes and DNS attributes in IP Flows.

```
Cisco NetFlow/IPFIX
    Version: 9
    Count: 1
    SysUptime: 382126128
  + Timestamp: Jun 30, 2014 00:02:55.000000000 EST
    FlowSequence: 2963
    SourceId: 0
  - FlowSet 1
      FlowSet Id: (Data) (257)
      FlowSet Length: 34
    - Flow 1
        SrcAddr: 10.0.0.8 (10.0.0.8)
        DstAddr: 172.16.0.100 (172.16.0.100)
        SrcPort: 4491
        DstPort: 80
        Protocol: 6
        TCP Flags: 0x02
        Octets: 144
        Packets: 3
      - [Duration: 8.892000000 seconds]
          StartTime: 382100.504000000 seconds
          EndTime: 382109.396000000 seconds
```

Figure 8: Sample flow record from generic template

The template in Figure 7 does not make use of the application layer data for detecting the botnets. We can also use additional filed for the generic template for high detection accuracy of specific bots. Figure 9 shows the generic template with additional fields related to packet size that can use for detecting specific bots such as Zeus by considering their unique C&C characteristics.

```
Exporter Format: NetFlow Version 9
Template ID    : 256
Source ID      : 0
Record Size    : 35
Template layout

|                Field                | Type | Offset | Size |
|-------------------------------------|------|--------|------|
| ipv4 source address                 |    8 |      0 |    4 |
| ipv4 destination address            |   12 |      4 |    4 |
| transport source-port               |    7 |      8 |    2 |
| transport destination-port          |   11 |     10 |    2 |
| ip protocol                         |    4 |     12 |    1 |
| transport tcp flags                 |    6 |     13 |    1 |
| ip length header                    |  189 |     14 |    1 |
| ip length payload                   |  204 |     15 |    2 |
| ip length total                     |  224 |     17 |    2 |
| counter bytes                       |    1 |     19 |    4 |
| counter packets                     |    2 |     23 |    4 |
| timestamp sys-uptime first          |   22 |     27 |    4 |
| timestamp sys-uptime last           |   21 |     31 |    4 |
```

Figure 9: Generic template with additional attributes

```
Cisco NetFlow/IPFIX
    Version: 9
    Count: 1
    SysUptime: 4008853000
  + Timestamp: Aug 10, 2014 23:28:22.000000000 EST
    FlowSequence: 612172
    SourceId: 0
  - FlowSet 1
      FlowSet Id: (Data) (256)
      FlowSet Length: 39
    - Flow 1
        SrcAddr: 10.0.0.10 (10.0.0.10)
        DstAddr: 172.16.0.100 (172.16.0.100)
        SrcPort: 1462
        DstPort: 80
        Protocol: 6
        TCP Flags: 0x02
        IP Header Length: 20
        IP Payload Length: 28
        IP Total Length: 48
        Octets: 144
        Packets: 3
      - [Duration: 9.076000000 seconds]
          StartTime: 4008827.440000000 seconds
          EndTime: 4008836.516000000 seconds
```

Figure 10: Sample flow data

Figure 10 shows the sample flow record that is captured by the extended template and forwarded to the flow collector.

We have tested the performance impact of the Cisco router, by network flows. Figure 11 shows the average (five runs) overhead on the router for capturing the traffic flows with fixed IP flows (NetFlow v5) and our generic template. Hence our generic template has minimal overhead on the routers and the detection accuracy is higher since only the attributes that are efficient for the bot detection are used in the template.

3.3 Discussion

Our model has several advantages for detecting a range of bots. Let us consider some of the advantages.

Most of the related techniques make use of fixed templates for the classification of the flows and/or can only detect specific bot families. Techniques, which make use of the fixed templates, can be applied to devices from specific vendors. Also the fixed templates are not efficient since many of the attributes that are used for the flow classification are not usable for detecting the bots. Furthermore it is not possible to extend these templates with efficient attributes for detecting the attacks. Hence the techniques, which make use of the fixed templates, have higher dataset overhead and are less efficient. Currently, there are several bot families such as IRC, HTTP, and peer-to-peer bots. Hence the techniques that can detect only specific bots are not much use.

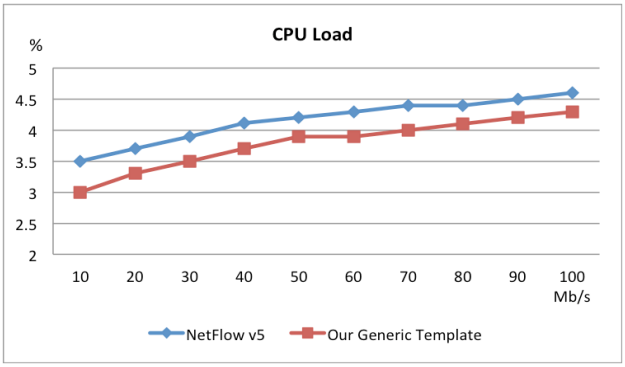

Figure 11: CPU Overhead

Our model makes use of only the attributes, which are efficient for detecting the bots for developing the templates. Since we make use of IPFIX for the specification of the template, it can be applied on the devices that belong to different vendors. Furthermore it has minimal dataset overhead since it only requires 30 bytes for a flow record and detecting a range of bot families. The overhead on the routers is also minimal compared to the fixed templates. Table 1 summarises the advantages of our model with related techniques.

4 Related Work

In this Section we will present some of the related techniques and compare them with our model.

BotSniffer (Gu, Zhang & Lee 2008) proposed network based anomaly detection techniques for detecting the bots. This system is capable of identifying C&C servers as well as infected hosts in a network. Detection is based on the fact that command and control communication within the same botnets most likely exhibit spatial-temporal correlation and similarity such as coordinated communication, propagation, DDoS and fake activities. However, this methodology limited to detect centralized botnets (IRC and HTTP).

Paper	Botnet Family								Other	
	HTTP	IRC	P2P	Generic	ICMP	IPv6	DNS	SMTP	Record Size (per record)	IPFIX compatible
Botfinder (2012)	✔	✔							48bytes	
Disclosure (2012)	✔	✔							48bytes	
Xiaonan Zang et al (2011)	✔	✔							48bytes	
Timothy Strayer (2008)		✔							48bytes	
Carl Livandas et al (2006)		✔							48bytes	
David Zhao et al (2012)			✔						48bytes	
Huihui Hou et al (2012)				✔					48bytes	
BotTrack (2011)			✔						48bytes	
Wernhuar Tarng et al (2011)			✔						whole packet	
BotSniffer (2008)	✔	✔							48bytes	
Our Method	✔	✔	✔	✔	✔	Potential	Potential	Potential	30bytes	✔

Table 1: Comparison with related techniques

Botfinder (Tegeler et al. 2012) proposed a methodology that senses bot in a network, using NetFlow v5 dataset. Botfinder leverage the discovery that C&C communication of a particular bot family trail specific regular pattern. This technique uses average time between start times of two subsequent flows in the trace, average duration of a connection, average number of bytes transferred to the destination, average number of packets transferred to the destination and Fourier transformation over the flow start times in the trace to detect botnet C&C activities. This approach also limited to detect centralized botnets (IRC and HTTP).

(Zhao et al. 2012) analyze NetFlow v5 characteristics to detect peer-to-peer botnet by introducing time windows for NetFlow v5 dataset. However, this approach has relied on the selection of correct size of the time window to cover activities of a bot. So they proposed two phases in their framework; training phase to find correct size for time window and detection phase to detect botnets from active flows. They have used 4 attributes from the NetFlow v5 dataset, which are source IP, source port, destination IP, destination port, L4 protocol. Then they have calculated 8 more attributes from NetFlow data that are, number of reconnects for a flow, number of flows from this address over the total number of flows generated per hour and attributes with in selected time window such as average payload packet length, variance of payload packet length, number of packets exchanged, number of packets exchanged per second, the size of the first packet, the average time between packets while examine behavior of peer to peer botnets such as Strom, Nugache and Waledac. This approach limited to peer-to-peer botnet and need to have complete network flow in order to effectively detect peer-to-peer bots.

(Zhao et al. 2013) examines the feasibility of detecting botnet activity without seeing a complete network flow by classifying behavior based on time intervals as extension to their previous study. Also the paper shows experimentally that it is possible to identify the presence of botnet activities with high accuracy even with very small time windows. Authors mainly addressed the limitation of payload inspection that resource intensive behavior, inability to deal with encryption and violation of privacy by using flow data to analysis. Detection method explained here was based on Network flow v5 while detecting real time botnet activities by inspecting the characteristics of these flows in small windows. This approach is also specific to peer-to-peer botnet detection.

The system called DICSLOSURE (Bilge et al. 2012) present a large-scale botnet (Centralized) detection method based on NetFlow v5 data and improved machine learning techniques. Authors identified unavailability of network data sets and terabits per seconds line speed. Authors consider following limitations of NetFlow in order to build their system. The NetFlow doesn't contain payload of the packet but aggregated metadata of the packet flows. Another limitation of the NetFlow is it can only consider one direction of the flow. NetFlow sampling used in large networks makes another limitation for malware detection. The system (DICSLOSURE) rely on NetFlow attributes; the source IP address, The destination IP address, Source port, Destination Port, Start and finish time of the flow, number of packets and number of bites transferred. Flow size, client access pattern and temporal behavior have been used to distinguish C&C communication over benign traffic.

(Strayer et al. 2008) constructed NetFlow v5 based botnet detection method for IRC bots. This technique first eliminates traffic that is unlikely to be a part of a botnet, classifies the remaining traffic into a group that likely to be part of a botnet, then correlates the likely traffic to find common communications patterns that would suggest the activity of a botnet. Strayer uses a five step filtering mechanism for filtering the traffic. We have already presented a detail discussion and comparison with this work in Section 2.2.3.

(Zang et al. 2011) introduced a fine flow classification method to detect centralized botnet using RTT (Round Trip Time) of the IP packets in NetFlow version 5 data set and K-mean clustering algorithms. Also authors highlighted that the difficulty of preparing universal system to detect all type of botnets. (Tarng et al. 2011) proposed a six-step methodology to identify peer-to-peer botnets by analysing network flow traffic and the payload characteristics. As the authors rely on payload characteristics this method not possible to be applied on high volume data networks. BotTrack (Francois et al. 2011) extend the Google's linkage analysis algorithm named PageRank, by adding clustering process based on DBSCAN algorithm in order to detect peer-to-peer botnet which are not generating large amount of communication traffic. NetFlow v5 data is used to define linkages in each

host and tracking communication patterns. Linkage analysis and clustering techniques apply together, to explore group of hosts showing similar behavioural patterns. This method is limited to detect peer-to-peer botnets.

From the above discussion and considering Table 1, it is clear that the related techniques make use of the fixed template available in Netflow v5 and can only detect specific bot families. Also high overheads are in their approaches in the aspect of data set size and CPU utilization in order to capture the IP flows. We have shown that our model can detect a range of bots and has minimal overhead.

Popular commercial systems such as Arbor-Peakflow, Radware-Defenceflow and Cisco Guard are able to detect botnets in attack phase (ex. DDOS), however our system can detect, bots during different phases in botnet life cycle (ex. infection, command and control, update and attack).

5 Conclusion:

In this paper we have proposed traffic analysis techniques for detection a range of bot families. Most of the related techniques make use of the fixed IP flows available in different products for detecting specific bot families. Our model makes use of IPFIX for designing a generic template to detect a range of bot families. We have also shown that generic template has minimal overhead on the routers. In the future work we will extend this model to deal with new types of botnets.

6 References

Andriesse, D, Rossow, C, Stone-Gross, B, Plohmann, D & Bos, H 2013, 'Highly resilient peer-to-peer botnets are here: An analysis of Gameover Zeus', paper presented to Malicious and Unwanted Software: "The Americas" (MALWARE), 2013 8th International Conference on, 22-24 Oct. 2013.

Antonakakis, M, Perdisci, R, Lee, W, Vasiloglou II, N & Dagon, D 2011, 'Detecting Malware Domains at the Upper DNS Hierarchy', paper presented to USENIX Security Symposium.

Bilge, L, Balzarotti, D, Robertson, W, Kirda, E & Kruegel, C 2012, 'DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis', paper presented to Annual Computer Security Applications Conference (ACSAC '12), Orlando, Florida USA.

Cisco Inc 2012, *Introduction to Cisco IOS® NetFlow*, <http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper.

Feily, M, Shahrestani, A & Ramadass, S 2009, 'A Survey of Botnet and Botnet Detection', paper presented to Emerging Security Information, Systems and Technologies, 2009. SECURWARE '09. Third International Conference on, 18-23 June 2009.

Francois, J, Wang, S, State, R & Engel, T 2011, *BotTrack: Tracking Botnets Using NetFlow and PageRank*.

Gu, G, Zhang, J & Lee, W 2008, 'BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic', paper presented to 15th Annual Network and Distributed System Security Symposium (NDSS'08), San Diego, CA.

Huawei Technologies Co Ltd 2012, *NetStream (Integrated) Technology White Paper*, viewed 1 enterprise.huawei.com/ilink/enenterprise/download/HW_201022.

IETF 2007, *IP Flow Information Export (ipfix)*, http://datatracker.ietf.org/wg/ipfix/documents/.

Juniper Networks Inc 2011, *Juniper Flow Monitoring*, <http://www.juniper.net/us/en/local/pdf/app-notes/3500204-en.pdf%3E.

NetSA, C 2014, *CERT NetSA Security Suite*, https://tools.netsa.cert.org/silk/.

Quittek, J, Zseby, T, Claise, B & Zander, S 2008, *Requirements for IP Flow Information Export (IPFIX)*, http://www.ietf.org/rfc/rfc3917.txt.

Sood, AK, Enbody, RJ & Bansal, R 2013, 'Dissecting SpyEye – Understanding the design of third generation botnets', *Computer Networks*, vol. 57, no. 2, pp. 436-50.

Strayer, WT, Lapsely, D, Walsh, R & Livadas, C 2008, 'Botnet Detection Based on Network Behavior', in W Lee, C Wang & D Dagon (eds), *Botnet Detection*, Springer US, vol. 36, pp. 1-24.

Tarng, W, Den, L-Z, Ou, K-L & Chen, M 2011, 'The analysis and identification of P2P botnet's traffic flows', *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 3, p. 138+.

Tegeler, F, Fu, X, Vigna, G & Kruegel, C 2012, *BotFinder: finding bots in network traffic without deep packet inspection*, ACM, Nice, France.

YouTube 2013, *Warbot Tutorial*, 06-May-2014, <https://http://www.youtube.com/watch?v=U8yrodbHg1Q%3E.

Zang, X, Tangpong, A, Kesidis, G & Miller, DJ 2011, *Botnet Detection Through Fine Flow Classification*, The Pennsylvania State University, University Park, PA, 16802.

Zhao, D, Traore, I, Ghorbani, A, Sayed, B, Saad, S & Lu, W 2012, 'Peer to Peer Botnet Detection Based on Flow Intervals', paper presented to 27th IFIP TC 11 Information Security and Privacy Conference, SEC 2012, Crete, Greece.

Zhao, D, Traore, I, Sayed, B, Lu, W, Saad, S, Ghorbani, A & Garant, D 2013, 'Botnet detection based on traffic behavior analysis and flow intervals', *Computers & Security*, vol. 39, Part A, no. 0, pp. 2-16.