BOTNET DETECTION IN VIRTUAL ENVIRONMENTS USING NETFLOW

Mark Graham
mark.graham@anglia.ac.uk

Adrian Winckles
adrian.winckles@anglia.ac.uk

Andrew Moore
andrew.moore@anglia.ac.uk

Department of Computing and Technology
Anglia Ruskin University
Cambridge, CB1 1PT

ABSTRACT

For both enterprises and service providers, the exponential growth of cloud and virtual infrastructures brings vast performance and financial benefits but this growth has undoubtedly introduced unforeseen problems in terms of new opportunities for malware and cybercrime to flourish. Botnets could be created entirely within the cloud using virtual resources, for a myriad of purposes including DDoS-as-a-Service.

This study has sought to determine whether distributed packet capture utilising mirroring technology or some form of sampling mechanism provides better performance for detecting cybercrime style activities within virtual environments. Recommendations are for a distributed monitoring technique which can provide end-to-end monitoring capabilities while minimising the performance impact on popular adoptions of cloud or virtual infrastructures. Investigations have concentrated on distributed monitoring techniques utilising virtual network switches, looking for a proof of concept demonstrator where sample Command & Control and Peer-to-Peer botnet activities can be detected utilising flow capture technologies such as NetFlow, sFlow or IPFIX.

This paper demonstrates how by inserting a monitoring function into a virtual or cloud architecture the capture and analysis of traffic parameters using NetFlow can be used to identify the presence of an HTTP-based Command & Control botnet.

KEY WORDS: NetFlow Protocol, C&C Botnet, Zeus Malware, Network Security, Virtual Environment

1.0 INTRODUCTION

A *bot* is an intelligent software application used to perform coordinated activities over the Internet. Whilst bots can be used to perform innocent, repetitive operations such as searching through quantities of data for web indexing or to locate a signal indicating the existence of life beyond our solar system, the majority of bots are created for malicious purposes. Malicious bots have become the most popular attack vector for modern malware. Bots work together to create a *botnet* of many thousands of infected hosts under the remote control of a human operator, or *botmaster*. This botnet serves as a single platform from which to launch massive co-ordinated attacks such as spam, distributed denial-of-service (DDoS), phishing and click-fraud.

Anti-virus software and Intrusion Detection Systems use signature definitions of known malware to identify malicious code entering a system. Every signature definition is unique to a specific malware variant, requiring a malware sample to first be acquired, reverse engineered, then the binary analysed for tell-tale code patterns that can be used to create that signature definition. Malware authors use a variety of mutation techniques to make minor changes to code whilst maintaining the attack profile, thus rendering the current signature definition useless in recognising the new modified malware. Signature-based detection has little effect on a botnet. Whilst it can recognise a botnet executable binary and disinfect a compromised machine, botnets typically comprise of thousands of bots, whereby removing a single bot has little impact on the overall botnet.

Individual bots within a botnet must periodically communicate with their Command and Control (C&C) server in order to receive updates or attack instructions. This communication traffic across a network or cloud can be used to indicate the presence of botnet activity. Traffic analysis techniques can be used to trace the C&C server(s) and takedown the botnet. Traffic analysis is typically reliant upon DNS records to detect bots searching for their C&C server (Arshad et al., 2011). To counter these detection techniques, malware authors have a range of evasion techniques to ensure their bots remain hidden. Techniques include DNS fast-fluxing (Stalmans and Irwin, 2011) to frequently and rapidly change the IP addresses corresponding to a domain name; and domain-fluxing where domain names are automatically generated for a corresponding IP address.

NetFlow has recently emerged as a new candidate for traffic analysis malware detection. Amini, Azmi and Araghizadeh (2014) were able to demonstrate that clustering of NetFlow data can be used for anomaly detection of C&C botnets across a simulated Local Area Network.

Virtualisation is now all pervasive across the field of information technology. Cloud computing adopts infrastructure virtualisation techniques to increase the utilisation of expensive hardware when offering services such as software-as-a-service (SaaS), platform-as-a-service (PaaS) and infrastructure-as-a-service (IaaS). More and more devices are becoming IP enabled as the world moves towards the Internet of Things. The emergence of the Crisis malware (OSX.Crisis and W32.Crisis) in August 2012 demonstrated that malicious botnets have started to move into virtual environments (Symantec, 2012). Crisis was the first malware to jump from a host operating system to a Virtual Machine (VM), actively seeking VMware virtual environments in which to propagate. Yet little research is being undertaken in order to understand signature-less botnet detection methods within virtual environments. Virtual environments have traditionally provided a safe, sandboxed environment in which to study the behaviour of malicious code. A common misconception is that malware is VM aware; able to detect the presence of a virtual environment and modify its behaviour accordingly, often refusing to execute. Lau and Svajcer (2010) showed that in reality as little as two percent of malware samples were proven to be VM aware.

This paper demonstrates that NetFlow can be used to detect the presence of C&C botnets within virtual environments. Section 2 of this paper discusses related work and defines the scope of this innovative work. In section 3, the general methodology and approach are described. Section 4 describes the results of our findings, which conclusions drawn in section 5.

2.0 RELATED WORK

A flow is a unidirectional stream of packets that pass through a network element and share a common set of attributes (Drago et al., 2011). One such flow technology is NetFlow, a network traffic monitoring and analysis protocol commonly used for network troubleshooting and performance improvements.

NetFlow was developed by Cisco System in 1996 (Kerr and Bruins, 2001) to provide enhancements over Simple Network Monitoring Protocol (SNMP) traps. SNMP is a polling protocol that queries a device every x minutes regardless of any changes in the environment. SNMP was not designed to carry large amounts of data and tends to carry usage overheads. NetFlow, like syslog, is a push technology requiring no device polling. However, the data format of syslog is unstructured which makes it both difficult and slow when used for reporting purposes. NetFlow version 9 forms the basis of the IPFIX standard defined in RFC7011 – 7016 (RFC 7011, 2013).

NetFlow data provides information about network conversations and behaviours. Today, most networking devices such as switches, routers and firewalls contain NetFlow probes that capture flow data and report this sampled data back to a collection server for analysis. NetFlow is highly structured with seven key fields: *source interface, type of service, source IP address, destination IP Address, source Layer-4 ports, destination Layer-4 ports and IP protocol* (Patterson, 2012). This makes NetFlow highly configurable, and as such flows can be used to capture almost anything at the network layer. When used to identify agents producing additional load on the network, NetFlow is effective in identifying unusual programs such as botnets (Amini, Azmi and Araghizadeh, 2014).

A drawback of using NetFlow as a data source is the vast amounts of data generated by the probes. Advanced analysis techniques such as data-mining are needed to make sense of the information contained within the NetFlow data. As described in section 1, bots must periodically communicate with their C&C server. Collecting and analysing this traffic data can reveal patterns that indicate the presence of botnet activity. Network traffic information such as ports, packet size, packet direction and conversation duration can be used to distinguish bot traffic from regular Internet traffic. In order to overcome the vast amounts of flow data being captured, two techniques are emerging to interpret collected NetFlow data: clustering and correlation. BotCloud (Francois et al., 2011) collected enormous quantities of NetFlow data. Dependency graphs were constructed from this data to identify infected hosts participating in the botnet, by correlating the destination IP address from node A with the source IP address from interlinked node B. MapReduce, a modified PageRank algorithm, was used to discover patterns within the dependency graphs, which were then clustered to hosts with similar patterns. DISCLOSURE (Bilge et al., 2012) used correlation to evaluate NetFlow record attributes i) flow size, ii) client access patterns and iii) temporal behaviour, to reliably distinguish C&C channels from innocent traffic in datasets in the magnitude of billions of flows per day. Amini, Azmi and Araghizadeh (2014) used data-mining combined with

clustering to analyse NetFlow data for anomaly detection of C&C botnets across a simulated Local Area Network. However no work has been found quantifying the success of this technique in real networks.

In summary, Botnet detection requires a different approach to detecting traditional malware. Signature-based detection can disinfect a device compromised by a bot. Removing one bot has a minor impact on the overall botnet which might comprise of thousands of compromised devices. Botnet takedown requires the analysis of bot communication traffic to identify and eradicate the C&C server(s). This can be done within networks and the cloud using a variety of techniques. Traditionally DNS records are used, but many DNS evasion techniques exist. NetFlow has emerged as a potential new candidate for traffic based detection. Little work has been done to develop these traffic analysis techniques for botnet eradication within virtual environments.

3.0 APPROACH

3.1 Zeus Bot

Our test network will be infected with Zeus bot. Zeus is a popular malware botnet that has been around since 2007. Zeus is typically used to steal banking information by man-in-the-browser, key logging and form grabbing. Despite its age there are still new active Zeus C&C servers being discovered daily (Zeus Tracker, 2014). One reason for the popularity of the Zeus bot is the easily accessible DIY construction kit, which allows someone with malicious intent to create, deploy and manage a Zeus botnet. The Zeus botnet created for this research was created from the Zeus crimewave toolkit v2.0.8.9.

When a device is infected with Zeus, the bot runs silently in the background giving no tell-tale signs of the compromise. As a C&C botnet, the Zeus bot must periodically communicate with its C&C server for updates, attack instructions or to report back with stolen information. With Zeus, this communication takes place over HTTP (TCP port 80). It is this communication traffic that this research intends to capture and analyse, in order to monitor the progression of Zeus across the virtual infrastructure.

Zeus bot can be detected via two categories of flow characteristics (Amini, Azmi and Araghizadeh, 2014):

- Static Characteristics: source and destination IP address, source and destination port numbers and protocols
- Dynamic Characteristics: packet event times, bytes per packets, periodic throughput samples, POST and GET.

3.2 Sampling Locations

NetFlow was defined in section 2 as a stream of packets that pass through a network element. Flow can be captured from any point in a network. An advantage of flow analysis over other mirroring or splicing techniques is that flow packets can be pushed to a collector without having to insert an additional device into the data stream. Therefore flow collection does not have a direct impact on the flow of traffic through the virtual environment.

Within a virtual environment there are several locations from which NetFlow data can be sampled:

- on a ***virtual interface*** - to collect potential malicious traffic exiting or entering a VM
- on a ***virtual switch*** - to collect potential malicious traffic communications between VMs
- on a ***hypervisor*** – to collect potential malicious traffic communication between network bridges connecting VMs

Figure 1 highlights the potential capture points within the test-bed network. Multiple capture points will be chosen across the virtual environment in order to ensure as much flow traffic as possible is captured thus reducing the risk of missing that vital flow stream that indicated botnet communication. NetFlow will be sampled at the following virtual interfaces (VI):

- 192.168.10.10 captures traffic entering and leaving the Zeus C&C VM
- 192.168.10.9 captures traffic entering and leaving the compromised victim VM
- 192.168.10.5 captures traffic destined for operating systems on the physical server hypervisor (192.168.10.2) and the flow collector (192.168.10.11)

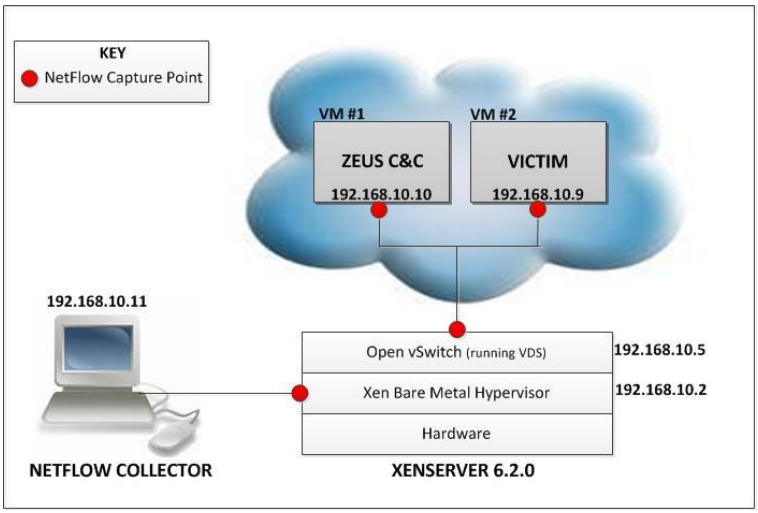

Figure 1 NetFlow capture points

3.3 Methodology

3.3.1 Physical and Virtual Network Architecture

Figure 1 shows the virtual and physical network topology of the test network. Citrix XenServer (Citrix Systems, Inc., 2014) was chosen as the hypervisor for the test virtual environment. XenServer is widely adopted as a commercial platform within cloud data centres and also has the advantage of open source licensing. If time permits, these monitoring functions will be tested within other vendor virtual environments such as VMware and Microsoft's Hyper-V in order to determine which aspects of the monitoring function are hypervisor architecture specific and which are generic.

XenServer 6.2.0 will be installed on a Linux server (192.168.10.2). A virtual network is created comprising of two virtual machines a) 192.168.10.10 will house the Zeus C&C server, b) 192.168.10.9 will be infected with the Zeus bot. The VMs are connected via Open vSwitch running Virtual Distributed Switch (192.168.10.5). The Distributed Switch (VDS) provides a centralised interface from which the configuration, monitoring and administration of VM switching for the entire virtual network can be controlled. Open vSwitch is configured to export NetFlow v5.

Connected to the XenServer is a flow collector (192.168.10.11). This Window-based PC will run Citrix XenCenter, which provides a simple real-time graphical display of flow activity across the VM (Figures 2 and 3) and is included within the XenServer application bundle.

3.3.2 Infection

The Zeus toolkit includes the Zeus Builder application, which is used to create the executable malware. This malware is transferred to the victim VM (192.168.10.9) and executed. Following execution of the malware, the bot can be seen and controlled by the C&C server in VM #1 (192.168.10.10) using the Zeus Control Panel application. Zeus botnet v2.0.8.9 uses a centralised C&C topology. All propagated bots need to communicate with the C&C server in VM #1 to receive instructions. Zeus communicates via HTTP on TCP port 80. The collectors are configured so as to monitor and collect all HTTP traffic.

3.3.3 Flow Collection Parameters

As discussed in section 2 NetFlow is highly structured with seven key fields: *source interface, type of service, source IP address, destination IP Address, source Layer-4 ports, destination Layer-4 ports and IP protocol.*

Amini, Azmi and Araghizadeh chose a seven field tuple to detect botnet activity in a simulated network:

a) Static characteristics: i) source/destination IP address, ii) source/destination port, iii) protocol
b) Dynamic characteristics: i) packet event times, ii) bytes per packet, iii) periodic throughput samples, iv) POST and GET

They chose to capture POST and GET as Zeus is an HTTP bot. Number of packets, bytes per packet, duration and POST/GET were captured for use in their flow clustering algorithm.

As this research is a proof of concept to demonstrate that botnets can be detected within virtual environments, the parameters captured in this test network differ slightly. Bytes per packet and POST/GET information will not be recorded. Instead TCP control parameters will be collected in order to recognise HTTP handshakes between the C&C and bot. In this research, similar tuple fields will be collected:

a) Static characteristics: i) source/destination IP address, ii) source/destination port, iii) protocol
b) Dynamic characteristics: i) packet event times, ii) TCP control parameters

4.0 RESULTS

Our research was conducted on a live physical network test-bed, hosting a virtual environment in which one VM contained Zeus C&C server and a second VM was infected with a Zeus bot. We were able to collect sufficient flow parameters in order to determine that the infection process and subsequent communication between C&C and bot could be detected.

Figures 2 and 3 graphically display the captured flow traffic over time. Spikes in the graphs represent different traffic streams captured over a 15 minute time period. Different protocols are visible as different colours. These different protocols are further identifiable within the flow records in Figure 4. Of primary interest is the HTTP (TCP 80) traffic which shows bot communication between IP addresses 192.168.10.10 and 192.168.10.9.

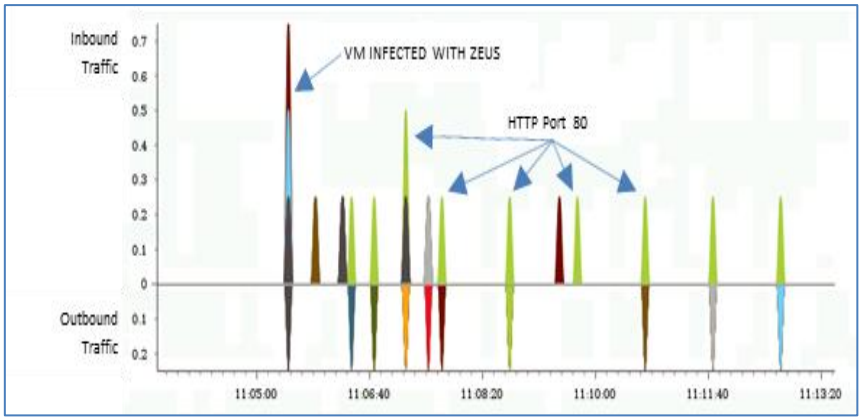

Figure 2 Behaviour pattern from victim VM infected with Zeus

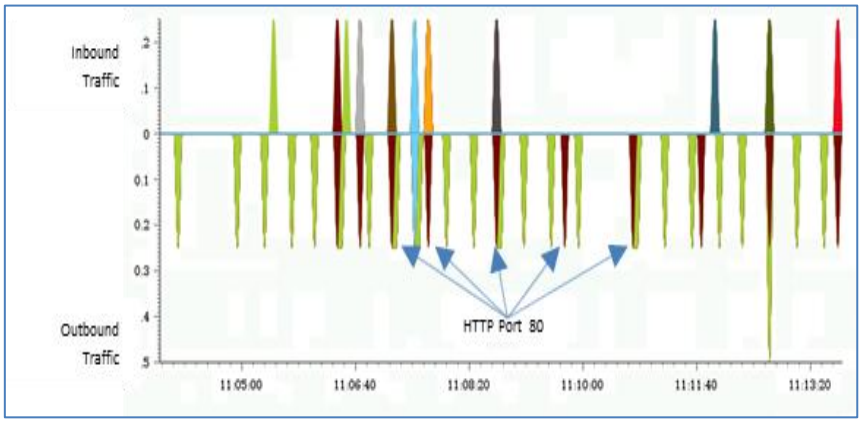

Figure 3 Behaviour pattern from VM containing Zeus CC

The infection process (Zeus infecting VM #2) is clearly visible in Figure 2, at 11:05:20, as a multi-coloured spike. Zeus bot was configured to communicate with its C&C server every minute. As such, HTTP spikes are visible every minute following the infection. These are labelled in Figure 2. Figure 3 shows the corresponding activity on the C&C server (in VM #1) after the infection of the victim. For each HTTP spike in Figure 2, a matching HTTP spike can be seen in Figure 3 as clear evidence of the bot communicating with the C&C server.

Flow parameter records are visible in Figure 4. Of particular note is the TCP control parameters captured within the flow traffic, which shows the TCP handshake as the bot communicates with the C&C server. At 12:20 12.889 the bot in VM#2 sends a HTTP keep alive to the C&C server – seen as a TCP SYN. The C&C server corresponds with the acknowledgement TCP ACK SYN.

StartTime	Flow Start	Flow End	Source IPv4	Dest. IPv4	Application	Source Port	Dest. Port	TCP Ctrl
10/06/2014 12:20	12.2889	12.2889	192.168.10.9	192.168.10.10	http (80 TCP)	1045	80	SYN
10/06/2014 12:20	12.2889	12.2889	192.168.10.10	192.168.10.9	http (80 TCP)	80	1045	ACK SYN
10/06/2014 12:20	12.2896	12.2896	192.168.10.11	224.0.0.252	llmnr (5355 UDP)	64424	5355	N/A
10/06/2014 12:20	12.2910	12.2910	192.168.10.11	224.0.0.252	llmnr (5355 UDP)	60730	5355	N/A
10/06/2014 12:20	12.2916	12.3047	192.168.10.2	192.168.10.11	iop (2055 UDP)	52547	2055	N/A
10/06/2014 12:20	12.2956	12.2957	192.168.10.2	192.168.10.5	https (443 TCP)	443	48719	ACK PSH
10/06/2014 12:20	12.2956	12.2957	192.168.10.5	192.168.10.2	https (443 TCP)	48719	443	ACK
10/06/2014 12:20	12.2957	12.2957	192.168.10.5	192.168.10.2	https (443 TCP)	46723	443	SYN
10/06/2014 12:20	12.2957	12.2957	192.168.10.2	192.168.10.5	https (443 TCP)	443	46723	ACK SYN
10/06/2014 12:21	12.3228	12.3438	192.168.10.2	192.168.10.11	iop (2055 UDP)	52547	2055	N/A
10/06/2014 12:21	12.3280	12.3309	192.168.10.11	192.168.10.255	netbios (137 UDP)	137	137	N/A
10/06/2014 12:21	12.3343	12.3343	192.168.10.9	192.168.10.255	netbios (138 UDP)	138	138	N/A
10/06/2014 12:21	12.3489	12.3489	192.168.10.10	192.168.10.9	http (80 TCP)	80	1046	ACK SYN
10/06/2014 12:21	12.3489	12.3489	192.168.10.9	192.168.10.10	http (80 TCP)	1046	80	SYN
10/06/2014 12:21	12.3498	12.3628	192.168.10.2	192.168.10.11	iop (2055 UDP)	52547	2055	N/A
10/06/2014 12:21	12.3500	12.3502	192.168.10.11	192.168.10.2	https (443 TCP)	50024	443	ACK PSH

Figure 4 Flow Records

This research gathered data from a live virtual network rather than using a network simulation. As with any live network an amount of background networking traffic is anticipated. Whilst this does not obfuscate the botnet communication, background traffic acts as noise when analysing the HTTP traffic. Future work will identify ways of distinguishing bot traffic from noise. Significant outbound background noise can be seen in Figure 3. These spikes were identified as "spytechphone" running on TCP port 8192; suggesting Sophos anti-virus was running within VM #1. Other captured background traffic is detailed in Figure 5.

HTTPS	TCP 443	XenServer (192.168.10.2) communicating with the collector (192.168.10.11)
OpenFlow	TCP 6632	XenServer (192.168.10.2) communicating with Open vSwitch (192.168.10.5)
OpenFlow	TCP 6633	XenServer (192.168.10.2) communicating with Open vSwitch (192.168.10.5)
Flow	UDP 2055	The collector was configured to capture flow traffic on UDP port 2055
NetBIOS	UDP 137	A consequence of running a Windows network

Figure 5 Background Traffic Protocols

The results seen in figures 2 & 3, namely the infection process and subsequent 1 minute communication between the bot and C&C, are in line with expected results extrapolated from Amini. Amini was able to see the same periodic HTTP pulse of bot communication, albeit over a different wavelength which would be due to a different setup in the bot executable.

5.0 CONCLUSION

This paper has demonstrated that NetFlow v5 can be used within a virtual environment to identify traffic pertaining to malicious botnets. The brief, regular communication between bots and their C&C server differentiates botnets from other network traffic. The traffic patterns identified within our captured flow data stream clearly show a bot located within a compromised virtual machine communicating over HTTP (TCP port 80) with its Zeus C&C situated within a different virtual machine.

A virtual environment is the ideal platform on which to host malicious botnet, as it enables a vast botnet to be created quickly and simply through the multiple cloning of infected VMs. Such platforms provide an environment for hosting malicious services such as DDoS-as-a-Service, which is now a reality. Virtualisation is an established building block in the provision of cloud-based service provider infrastructures. The Crisis malware (OSX.Crisis and W32.Crisis) targeted virtual environments. Crisis is anticipated as the first malware to emerge in a currently un-tapped attack vector of virtualisation. The need to protect virtual environments as we move toward the Internet of Things and Smart Cities is clear.

5.1 Future Work

This proof-of-concept virtual environment will now be expanded to create a fully loaded test-bed mirroring real world virtual networks. Various placements of the monitoring function together with open source flow analysis tools will be contrasted for their success in gathering sufficient evidence to decipher the location of the C&C server(s) and hence enabling the takedown of the botnet.

REFERENCES

Amini, P., Azmi, R. and Araghizadeh, M., 2014. Botnet Detection using NetFlow and Clustering. *Advances in Computer Science: an International Journal*, 3 (2), pp.139-149.

Arshad S., Abbaspour M., Kharrazi M. and Sanatkar H., 2011. An Anomaly-based Botnet Detection Approach for Identifying Stealthy Botnets. *International Conference on Computer Applications and Industrial Electronics 2011.* IEEE, pp. 564 – 569.

Bilge L., Balzarroit D., Robertson W., Kirda E. and Kruegle C., 2012. Disclosure: Detecting Botnet Command and Control Servers through Large-Scale NetFlow Analysis.*28th Annual Computer Security Applications Conference, New York, 2012*. ACM, pp. 129-138.

Citrix Systems, Inc., 2014. XenServer (6.2.0). [computer program] Citrix Systems. Available at: <http://xenserver.org/overview-xenserver-open-source-virtualization/download.html> [Accessed 12 February 2014].

Drago, I., Barbosa R., Sadre, R., Pras A. and Schonwalder J.. 2011. Report of the Second Workshop on the Usage of NetFlow/IPFIX in Network Management. *Journal of Network and Systems Management, 19(2) 2011.* Springer, pp. 298-304

Francois, J., Wang, S., Bronzi, W., State, R., and Engel, T., 2011. BotCloud: detecting botnets using MapReduce. *International Workshop on Information Forensics and Security (WIFS) 2011.* IEEE, pp. 1-6.

Kerr, D., Bruins, B., Cisco Systems, Inc., 2001. *Network Flow switching and flow data export*. U.S. Pat. US6243667 B1.

Lau, B., & Svajcer, V.,2010. Measuring virtual machine detection in malware using DSD tracer. *Journal in Computer Virology*, *6*(3).Springer, pp. 181-195.

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

[RFC 7011] *IPFIX: IP Flow Information Export Protocol for the Exchange of Flow Information,* IETF, September 2013, [online] Available at: < http://www.ietf.org/rfc/rfc7011.txt> [Accessed 8 April 2014].

Stalmans, E. and Irwin, B., 2011. A framework for DNS based detection and mitigation of malware infections on a network. *Information Security South Africa (ISSA), 2011.* IEEE, pp. 1-8.

Symantec, 2012. Crisis for Windows Sneaks onto Virtual Machines, [online] Available at: <http://www.symantec.com/connect/blogs/crisis-windows-sneaks-virtual-machines> [Accessed 1 June 2014]

Zeus Tracker, 2014. Zeus Tracker, [online] Available at: <https://zeustracker.abuse.ch/monitor.php> [Accessed 10 June 2014]