Towards Malicious Network Activity Mitigation through Subnet Reputation Analysis

Conference Paper · September 2016

CITATIONS
2

READS
121

2 authors:

Alan Herbert
Rhodes University
17 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Barry Irwin
Noroff University College
183 PUBLICATIONS **877** CITATIONS
SEE PROFILE

Towards Malicious Network Activity Mitigation through Subnet Reputation Analysis

Alan Herbert[1], Barry Irwin[2]
Department of Computer Science, Rhodes University, P.O. Box 94, Grahamstown 6140, South Africa
[1]g09h1151@campus.ru.ac.za
[2]b.irwin@ru.ac.za

***Abstract*—Analysis technologies that focus on partial packet rather than full packet analysis have shown promise in detection of malicious activity on networks. NetFlow is one such emergent protocol that is used to log network flows through summarizing key features of them. These logs can then be exported to external NetFlow sinks and proper configuration can see effective bandwidth bottleneck mitigation occurring on networks. Furthermore, each NetFlow source node is configurable with its own unique ID number. This feature enables a system that knows where a NetFlow source node ID number resides physically to say which network flows are occurring from which physical locations irrespective of the IP addresses involved in these network flows.**

The backend of the Bolvedere system consists of multiple processor modules that run concurrently in order to discern malicious activity out of NetFlow logs. This research analyses the findings of these concurrent modules and couples these findings with physical geolocations based on the NetFlow source node ID rather than the geolocation of the network flow's IP addresses which could be spoofed; as is common practice in malicious network activity. These coupled results are then used at a later stage as a mechanism to better point out the geolocation in which malicious network activity is being sourced. Geolocation based mitigation can then occur more effectively from this point.

***Index Terms*—Digital forensics, Internet security, Network security, Stream processing**

I. Introduction

This research works along side with the module repertoire that is compatible with the Bolvedere platform that is currently in development by the authors of this paper. The Bolvedere [1] platform is a highly adaptable and scalable NetFlow processor intended for distributed identification of malicious network activity. All modules developed for this platform run concurrently, be it remote to the Bolvedere host system or within it. The implementation brought forward in this research is in the context of development of a new module to run on the Bolvedere system.

A. Problem Statement

Latent security in large scale networks is an ever increasing requirement of modern networks. The need for traffic monitoring to protect users from malicious attacks, as well as prevention of systems within a network being able to produce a malicious attack, are some basic requirements of a secure network. If a network is unable to ensure this, systems on that network could be exploited and then used within botnets to proxy malicious data and other malicious activities on the Internet [2]. This as a whole is detrimental to data flow on the Internet and can disrupt secure networks attached the Internet through malicious network activity sourced from an insecure network at a secure one.

Due to the sheer volume of network traffic generated on modern networks, automated systems are a must for effective network traffic monitoring. These systems should be able to provide feedback to concurrent systems in order to better mitigate malicious activities targeted at the network, or within the network itself.

B. Research Goal

This research in its entirety aims to develop a modular platform for which modules that process network flow traffic can interface in order to discern a set of known occurrences on a network. Adaptability is taken into account in the form of resources available for such a system to be run and thus Bolvedere should be able to execute itself on a single host machine that can run a Linux OS (Operating System) as well as provide the ability to scale out over multiple threads, multiple processes, multiple processors and multiple separate physical hosts. This also includes support for multiple programming languages as well as hardware technologies such as GPUs (Graphical Processor Unit) and FPGAs (Field Programmable Gate Array). This scalability ensure the ability for Bolvedere to take on the task of Internet level network flow discernment as to a connection being malicious or legitimate [1].

C. Structure

This paper outlines and then shows how the implementation of a subsystem within Bolvedere can generate a reputation score for a network located on the Internet for each malicious activity type based on Bolvedere module findings. After a literature review of technologies used in this research in Section II, the strength of this implementation is then shown to arise from the use of a NetFlow source node ID number to place network flows at a specific physical world location instead of the use of IP addresses which could be spoofed; this is discussed in more detail in Section III. The effectiveness of this implementation in terms of attack mitigation is questioned in Section IV and conclusions are then drawn in Section V.

II. Literature Review

A. NetFlow[1]

The NetFlow protocol is best described as a means of logging network flows that pass through a flow monitoring device in a communication pair's route. A flow is defined by a connection and communication between a host and any other host, multicast group, or broadcast domain in the form of a sequence of packets [3]. A flow monitor using the NetFlow protocol can collect fields out of these communications, write them into predefined fields (restricted either by protocol version or through a known template) and then transmit them to a logging host for analysis or storage [4]. There have been multiple versions of NetFlow with wide-spread support over multiple firewall and routing devices on the Internet.

The need to update the NetFlow protocol over the years arose from multiple factors. First, the addition of IPv6 (Internet Protocol version 6) [5] that was brought about by the IP address exhaustion [6] of the IPv4 (Internet Protocol version 4) [7] address space required amendments to be added to the NetFlow protocol. Furthermore, the need for better use of network resources grew as the amount of traffic passing through flow monitor points increased. Finally, the requirement to adapt these records to one's needs gave way to updating NetFlow to give users the ability to break out of the predefined logging fields determined by older versions of NetFlow into a dynamic space that allows for field collection through a predefined and distributed template [4]. This also allowed users to log any protocols defined at a later stage.

Major updates to the original NetFlow standard have included the addition of new fields and further standardisation to NetFlow with version 5 of the protocol. This allowed for logging subnet masks and AS (Autonomous System) numbers [8]. Version 8 saw the inclusion of record aggregation that was first defined in version 5 [9]. More recently, version 9 continued to build on the freedom brought forth by version 8 through the addition of the template packet to the NetFlow protocol. This template packet allowed one to define the fields to be logged in a record and the order in which they are logged from a flow [4].

These templates are coupled with template identification numbers that allow for multiple templates to be used and are defined by the 2-byte-long template identification field within the NetFlow protocol. Although IDs 0 through to 255 are reserved for use by specific flow templates, template identification numbers 256 through to 65535 are available for public use; a fairly large template count. This template count further extends the memory requirements of these NetFlow devices and most devices supporting NetFlow version 9 limit the number of templates that can be stored to a count far less than the available 65280 [10]. Allowing for templates to define what should be logged and the order in which it should be done does have draw backs. These drawbacks are found in memory and performance, as the device receiving and using the template has to store and identify it for later use.

[1]This text appears in other works by the authors of this paper [1].

B. Packet Source IP Geolocation and Dealing with Spoofing

IP addresses on the Internet are assigned by blocks called subnets (subnetworks). These subnets are allocated to LIRs (Local Internet Registries) in the world and thus one can use the source IP address of a network flow in order to place where a network flow originated from in the world through IP address comparison to these registries [11]. This information can be used to regulate network connections through generation of firewall rules in order to allow, prevent or modify connections to certain IP addresses [12].

Blocking of an IP address or range of IP addresses may seem like an appropriate way in which to mitigate an attack from a known source IP address however it is not [13]. Commodity PCs (Personal Computer) can generate network packets in which all bytes within it are defined by the user. This is referred to as packet spoofing and is performed by modification of the source IP address bytes of a network packet to represent that of an IP address not belonging to that system. Furthermore, one can randomly generate these address for every packet that system transmits. This means that one cannot just block an IP address or set of IP addresses and be done with it; this is a very tedious problem to solve. Spoofing in this manner is typically used in the spectrum of DoS (Denial of Service) attacks [14].

There are defensive mechanisms in place for dealing with attacks that use IP address spoofing. The first and most inconvenient is to change one's systems static IP address/addresses. If the malicious system is set to attack a specified IP address, or works on the system of resolve FQDN (Fully Qualified Domain Name) once and then target the given IP address, then this would allow the target to mitigate this form of implemented attack [15]. This will work until the attacking system updates the IP in which it is targeting through administration or by resolving the target FQDN again.

Simply knowing who should be connecting to a system can be invaluable information to prevent random IP source address attacks. Creation of a whitelist containing only the subnets and IP address that should have access to your system is a method that can filter out a lot of unwanted connections. This also makes sense in business planning as there are many businesses that only do business with a set part of the world [16]. This means that one can include these whitelists into the business model of one's company.

If one does not have a deep understanding of how networks work, such as a small company that can't afford on-site system administration, one can employ a third-party company in which one's service is relayed through. This third-party company, like CloudFlare [17], can provide mitigation of malicious network activity on your behalf in order to keep one's services available and secure.

C. Use of NetFlow Node Source ID in Packet Source IP Spoofing Detection

As part of the NetFlow version 9 and IPFIX protocols [10], each NetFlow source node is allocated an ID number. If one were to ensure these IDs were unique and keep a record of

where each node is physically located, one would now have a method of detecting where certain types of network traffic are sourced from, be it within a private network or a public network, without the need for an IP address [18]. This is useful in locating where a network flow is physically sourced from when considering malicious activity that uses IP address spoofing as discussed in Section II-B.

To elaborate with an example consider a simple DoS attack on a target system. If the DoS attack was created in such a way that it spoofed traffic with sources from around the world, in order to prevent communications from the attack one would have to effectively block communication from the entire world; not very useful at all. Now if one had access to NetFlow sources from around the world and knew which NetFlow source ID numbers were allocated to each specific location in which a NetFlow source node was placed, one could use this information to better mitigate this example attack. From the logs generated from these NetFlow source nodes, one could identify which flows were destined to their service. Coupling this information with the NetFlow source ID number which each NetFlow log is received with, and assuming one can detect which logged flows are associated with the DoS attack, one could tell which locations in the world the attack is being generated from. From this point one could perform more effective attack mitigation.

III. Implementation

This research implements an Internet based reputation system that collects information based on what malicious network activities are sourced from where in the world. The world location for this implementation will be based on the the NetFlow source node ID number and known location of that ID number's placement in the world. Using this information, coupled with the NetFlow field which records which physical port the communication was received on, this implementation can tell if a network flow is being sourced onto the Internet, or sunk from it.

The next step in this implementation is to amalgamate the aforementioned information with detected malicious activities. This implementations purpose is not to detect malicious network flows and as this job falls onto the NetFlow processor modules of Bolvedere. Referring to Figure 1 one can see an overview of the entirety of the Bolvedere system. To better understand the logic flow of the Bolvedere system, the function of each subsystem is listed below:

- External Network: This is where NetFlow logs are received from into the Bolvedere system.
- Collectors: Dedicated hardware based NetFlow record collectors that re-order and filter flow records as to how the Publisher can optimally process these logs [1].
- Publishers: CPU based NetFlow filter processing component that receives records in requested order from hardware collectors in order to optimally process NetFlow logs and distribute these logs to connected Processors using ZMQ (Zero Message Queue).
- Processors: Modular sub-processes that discern processed NetFlow logs in order to detect malicious network activity. These sub-processes run concurrently with each other and due to the functionality of ZMQ, these modules can run within the same host system or on separate physical hosts entirely.
- Reputation System: This is the focus of this research that collects malicious activity detected by the processors of this system and couples this information with real world physical locations in order to better place where certain malicious network activities are physically sourced.

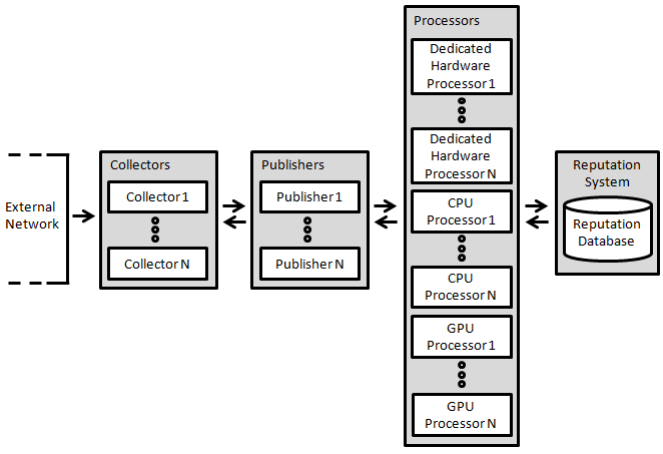

Fig. 1: Bolvedere System Overview

As mentioned, this research focuses on the implementation of a malicious network reputation system based on geographic location. This implementation will collect detected malicious network flows based on the results of the Bolvedere processor modules and couple these occurrences with the geographic location based of the NetFlow source node ID's geographic location from which the network flow was recorded. This will allow this implementation to build up a reputation database over time. In the event of a malicious attack on a system or service, this database can then be used to better pinpoint which physical location the attacker resides irrespective of source IP address which could be spoofed.

A. Tools Used

Consideration towards system bottlenecks has to be made when considering input data from multiple sources. These bottlenecks were identified to be the network component and the means in which the received data from the network is stored to a long-term medium. As ZMQ was already used for publishing NetFlow logs to the processor modules of Bolvedere, ZMQ was again utilized to pass information from the processor modules to the reputation system.

ZMQ is library that was developed to manage broadcast groups within networks. It is supported by over 30 languages and in the Bolvedere system this allows a module to use the language that is best for the task at hand. The configurations used in the Bolvedere system follows a "publisher subscriber" scheme between the publisher and processor subsystems, and a "push pull" configuration between processors and the reputation system. The "publisher subscriber" scheme allows the processors modules to only connect to the publishers which

publish NetFlows relevant to what the module is trying to achieve. The "push pull" scheme allows the processors to push results to the reputation system which then pulls these results and processes them before storage. Furthermore, the ZMQ transport layer can be set to in-process, inter-process, TCP and multicast modes to allow for communications between process within a single physical host, or between processes on separate physical hosts [19].

For record storage a SQL (Structured Query Language) database implementation was used. These database implementations use a common language and only vary in how their backend works. These backends vary from file access on disk (SQLite [20]) to complete in memory solutions (MemSQL [21]). One should base the implementation used on what the requirements of the system are and on what resources are available to one for use in the system. As these implementations all use one well defined language, swapping out the SQL implementation used is an easy process when the requirements of the system and/or resource availability changes.

B. Configuration

Configuration of this reputation system within Bolvedere is simply performed through specifying a database in which to use and then starting the pull interface on the ZMQ handle within the implementation. After this interface is successfully created, the Bolvedere processors are notified that the reputation system is ready to receive detected malicious network flows. From here system logic flow is as follows:

1) A NetFlow data record arrives at collector and is filtered and re-ordered as to what the publisher requires for optimal process throughput.
2) Publisher runs CPU based filters before publishing to processor modules.
3) Concurrent processor modules receive these processed NetFlow logs and discern whether a network flow contains malicious network activity or not. Any malicious activity is then pushed to the reputation system.
4) The reputation system pulls results from the processors and couples the information with the geographic information gathered by referencing to the NetFlow source ID received in the original NetFlow log.

IV. Results

This research sought to collect findings produced by processor modules running as part of the Bolvedere system and couple these findings with real world geographic locations based on the NetFlow node ID in which the network flows were generated from. As these findings are intended to be stored for use at a later point, this testing will be broken into two parts. The first is generation of malicious network activity in which the Bolvedere modules generate findings for the reputation system to store, and the second being the accuracy of these results that are stored and how they can be applied.

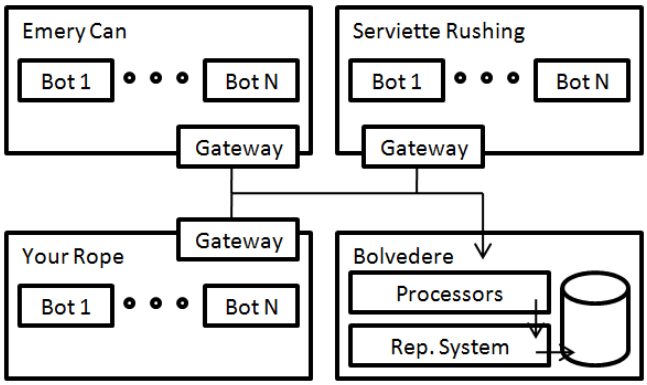

Fig. 2: Testing Environment Overview

A. Environment

These tests were all performed in a virtual environment which allowed the creation of disjoint networks that could then be connect through common gateways. This was used to represent an Internet like structure where a network with connections between Internet service providers and/or telecommunication companies would exist. The configuration of this can be referred to in Figure 2. To better explain objects and logic flow in this figure one can refer to the descriptions below:

- Bot: This refers to one of two automated system written by the authors of this paper. The one form of system generates legitimate network traffic through HTTP [22], ICMP [23], TCP [24] and UDP [25] requests. The second form generates malicious network activity through the use of Metasploit [26].
- Gateway: These are virtual systems that allow access to other virtual networks within the virtual environment. These gateway systems also generate NetFlow logs using softflowd [27] and thus act as NetFlow source nodes. Each of these softflowd processes running on the separate gateways has been recompiled to use a unique source ID.
- Bolvedere: This is the NetFlow sink of the virtual environment. All gateways send their NetFlow logs to the Bolvedere system which then process these logs to detect malicious activity before handing these findings to the reputation system.
- Network 1, 2 and 3: These are reference names of each virtual network in which the bots reside. These names exists purely for ease of referral in the results text.

B. Runtime Malicious Activity Collection

The purpose of the initial test is to gather malicious network flows from different networks and then say which network the activity originated from using the NetFlow source ID. To achieve this each of the three virtual network were assigned certain malicious tasks in which it would perform along with how often it should perform each. This would allow for checking accuracy at a later point as to the reputation score of each network according to the attacks performed; how likely a network was to perform a certain form of malicious network flow. A distribution of the malicious network tasks

TABLE I: Virtual Network IDs and Malicious Activity by Percentage

Name	ID	ms08_067_vnc	ms08_067_shell	samba_symlink	unreal_ircd_3281	ntp_mon_list
Network 1	100	40	40	10	0	0
Network 2	200	10	10	0	30	50
Network 3	300	0	0	90	0	0

TABLE II: Virtual Network Malicious Reputation Scores out of 100

Name	ID	ms08_067_vnc	ms08_067_shell	samba_symlink	unreal_ircd_3281	ntp_mon_list
Network 1	100	82.7	79.9	7.8	0	0
Network 2	200	17.3	20.1	0	100	100
Network 3	300	0	0	92.2	0	0

performed by each network and NetFlow source ID number can be referred to in Table I.

The results displayed by the reputations system were printed to terminal at runtime of this system and can be referred to in Listing 1. The format of the output is made up of the three key pieces of information, however there is more information stored in the reputation system's database that will be discussed in Section IV-C. The format is as follows:

- Src ID: NetFlow source node ID number in which the malicious flow record was sourced from.
- Loc: Network name that the Netflow source node ID is recorded as.
- Atk Type: The name of the type of attack that was detected in the malicious flow by a Bolvedere processor module.

Listing 1: Terminal Output of Reputation System

```
...

[Src ID:100, Loc:Network 1,
 Atk Type:ms08_067_shell]

[Src ID:100, Loc:Network 1,
 Atk Type:ms08_067_vnc]

[Src ID:100, Loc:Network 1,
 Atk Type:ms08_067_shell]

[Src ID:300, Loc:Network 3,
 Atk Type:samba_symlink]

[Src ID:200, Loc:Network 2,
 Atk Type:unreal_ircd_3281]

[Src ID:300, Loc:Network 3,
 Atk Type:samba_symlink]

[Src ID:200, Loc:Network 2,
 Atk Type:ntp_mon_list]

...
```

Results were collected over 1503 malicious network flows and 11312 non-malicious network flows. These results, again referring to the snippet in Listing 1[2], show the attacks that were assigned to the malicious bots are identified by the Bolvedere processor modules. Furthermore, the reputation system put in place to bind these findings to a set location also shows correct output for the test inputs provided to the system as a whole; this being the Bolvedere system and the reputation system.

C. Reputation Storage Results

The records of the 1503 malicious network flows that were recorded by the reputation system in Section IV-B will now be verified through processing of these records in this section. As the reputation of each network for all known attack findings given to this reputation system are generated at runtime, this section will simply look at these reputation scores and weigh them up against the known input testing to the Bolvedere system. The higher the score the more likely the network is to source the respective malicious attack.

The first positive that the results in Table II reflects is that the networks assigned not to perform a form of malicious attack in Table I do not score any points for those attacks. The next point to notice is that the *unreal_ircd_3281* and *ntp_mon_list* attacks that were only assigned to Network 2 score 100 for each of these malicious attacks in the Network 2's reputation. This is good as Network 2 is the only network that produces these attacks and so if these attacks occur, Network 2 is the only network that these attacks have been sourced from before and so is the most likely candidate; hence the 100 point rating towards this network.

Finally, the attacks that were assigned to multiple networks show a distribution of reputation points between the assigned network in accordance to the ratio assigned to each network. The error in points is due to the randomness introduced in the malicious bots when choosing which attack to perform. These results show that the reputation system does correctly store and assign a reputation based on type of attack to a network location. Furthermore, these results can be used in order to better point out which network a malicious attack is

[2]The size of the snippet is due to the space provided in this paper.

most likely to be sourced from and place that network flow at a known physical location irrespective of the source IP address.

V. Conclusion

This research aimed to determine whether there was an alternate method of identifying which network an attack originated from irrespective of source IP. Using the NetFlow protocol and Bolvedere system this research designed and implemented a reputation system in which the NetFlow source node ID numbers could be used in order to place a network flow at given point in the world. After testing this reputation system against 1503 malicious network flows detected by the Bolvedere processor modules, this reputation system showed that it could successfully give intelligence into where an attack of a set type was most likely to originate from. Given these results, this research was deemed successful and further Bolvedere processor module support will continue in the future work of this reputation system. Further iterations of this implementation seek to provide more expert information as well as a better interface in which to represent findings from the Bolvedere system.

Acknowledgement

This work was undertaken in the Distributed Multimedia CoE at Rhodes University, with financial support from Telkom SA, Tellabs, Easttel, Bright Ideas 39, THRIP and NRF SA (UID 75107). The authors acknowledge that opinions, findings and conclusions or recommendations expressed here are those of the author(s) and that none of the above mentioned sponsors accept liability whatsoever in this regard.

The authors also wish to acknowledge the support of the Council for Scientific and Industrial Research (CSIR).

References

[1] A. Herbert and B. Irwin, "Adaptable exploit detection through scalable netflow analysis," in *15th Annual Information Security for South Africa Conference*, 2016.

[2] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006, pp. 41–52.

[3] D. R. Kerr and B. L. Bruins, "Network flow switching and flow data export," Jun. 5 2001, uS Patent 6,243,667.

[4] B. Claise, "Cisco systems NetFlow services export version 9," *IEEE Networking Group RFC*, 2004.

[5] S. E. Deering, "RFC 2460: Internet Protocol, version 6 (IPv6) specificat," 1998.

[6] G. Huston. (2014, February) IPv4 Address Report. Accessed 6th February 2014. [Online]. Available: http://www.potaroo.net/tools/ipv4/index.html

[7] J. Postel, "RFC 791: Internet Protocol," IETF, Tech. Rep., 1981.

[8] G. Huston. (2006) NetFlow Packet Version 5 (V5). Cisco Systems, Inc. Accessed 13th February 2015. [Online]. Available: http://netflow.caligare.com/netflow_v5.htm

[9] ——. (2006) NetFlow Packet Version 8 (V8). Cisco Systems, Inc. Accessed 13th February 2015. [Online]. Available: http://netflow.caligare.com/netflow_v8.htm

[10] Cisco. (2003) NetFlow V9 Export Format. Cisco Systems, Inc. Accessed 13th February 2015. [Online]. Available: http://tinyurl.com/ond3pe8

[11] R. Housely, J. Curran, G. Huston, and D. Conrad, "Rfc 7020: The internet numbers registry system," 2013.

[12] K. F. King *et al.*, "Geolocation and federalism on the internet: Cutting internet gamblings gordian knot," *Colum. Sci. & Tech. L. Rev.*, vol. 11, pp. 41–58, 2010.

[13] P. C. Johnson, A. Kapadia, P. P. Tsang, and S. W. Smith, "Nymble: Anonymous ip-address blocking," in *Privacy Enhancing Technologies*. Springer, 2007, pp. 113–133.

[14] R. M. Needham, "Denial of service," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*. ACM, 1993, pp. 151–153.

[15] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: classification and state-of-the-art," *Computer Networks*, vol. 44, no. 5, pp. 643–666, 2004.

[16] A. Srivastava and J. Giffin, "Tamper-resistant, application-aware blocking of malicious network connections," in *Recent Advances in Intrusion Detection*. Springer, 2008, pp. 39–58.

[17] CloudFlare, Inc. (2016) Cloudflare protects and accelerates any website online. http://tinyurl.com/zawehua. Accessed 19th May 2016.

[18] Cisco. (2003) NetFlow Version 9 Flow-Record Format. Cisco Systems, Inc. Accessed 19th May 2016. [Online]. Available: http://tinyurl.com/24jvyz

[19] P. Hintjens, *ZeroMQ: Messaging for Many Applications*. " O'Reilly Media, Inc.", 2013.

[20] SQLite Consortium. (2016) Sqlite: Small. fast. reliable. choose any three. Accessed 30th April 2016. [Online]. Available: https://www.sqlite.org/

[21] MemSQL Inc. (2016) Make every moment work for you. Accessed 30th April 2016. [Online]. Available: http://www.memsql.com/

[22] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Rfc 2616, hypertext transfer protocol–http/1.1, 1999," *URL http://www. rfc. net/rfc2616. html*, 2009.

[23] J. Postel *et al.*, "Rfc 792: Internet control message protocol," *InterNet Network Working Group*, 1981.

[24] J. Postel. (1977, August) Comments on internet protocol and tcp. http://www.postel.org/ien/txt/ien2.txt. Accessed 1 August 2015. [Online]. Available: http://www.postel.org/ien/txt/ien2.txt

[25] ——, "Rfc 768: User datagram protocol, august 1980," *Status: Standard*, 1980.

[26] D. Maynor, *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Elsevier, 2011.

[27] D. Miller. (2016) Softflowd. Mindrot. Accessed 30th April 2016. [Online]. Available: http://www.mindrot.org/projects/softflowd/

Alan Herbert Currently studying for a PhD in Computer Science at Rhodes University supervised under Prof. Barry Irwin. His research interests base themselves heavily in hardware development, networks and network security.

Barry Irwin Heads the Security and Networks Research Group within the Computer Science Department at Rhodes University. His research interest encompass passive traffic analysis, network security, and national level cyber defence.