An Exploration of Geolocation and Traffic Visualisation Using Network Flows

Sean Pennefather and Barry Irwin
Department of Computer Science
Rhodes University
Grahamstown 6140
Email: g10p0016@campus.ru.ac.za, b.irwin@ru.ac.za

***Abstract*—A network flow is a data record that represents characteristics associated with a unidirectional stream of packets transmitted between two hosts using an IP layer protocol. As a network flow only represents statistics relating to the data transferred in the stream, the effectiveness of utilizing network flows for traffic visualization to aid in cyber defense is not immediately apparent and needs further exploration. The goal of this research is to explore the use of network flows for data visualization and geolocation.**

A prototype system capable of collecting network flows exported using the NetFlow version 9 protocol was designed and implemented as part of this research to aid in the exploration. This prototype system processes the collected flow records and renders the geolocated results on an web based interactive map.

Using conformance testing it is shown that the prototype system is capable of collecting network flows and generating geolocated flow events withing 50 milliseconds of receiving the raw flow records on the test platform. The system also provides functionality for the generation of heatmaps and tools for replaying flow events from the client browser for further visual analysis. A reporter tool has also been developed to produce monthly reports on the collected network flows.

I. Introduction

Network flow processing has the potential to allow for a large reduction in the volume of data to be processed by monitoring systems when compared to traditional packet processing counterparts. The reason for this reduction in volume is that a network flow is a single record that represents the characteristics associated with an instance of communication between two hosts using an IP layer protocol [1]. A flow record does not record the actual data transferred and as a result, the record size is only dependent on the number of characteristics the record must report on rather than the number of packets transferred during the lifetime of the connection.

This allows network flows to be used to reduce the volume of data that must be processed. This reduction comes at the cost of not recording the actual content of the packets that make up the connection which are required by systems that employ packet analysis techniques as part of processing [2]. Because of this reduction in resolution, the effectiveness of utilizing network flows for traffic visualization to aid in cyber defense is not immediately apparent and needs further exploration.

In order to explore the feasibility of Network Flows in the above applications, we have developed a system that is capable of performing both geolocation and traffic visualization. The visualization is achieved by generating heatmaps and monthly reports. This system reads in raw Network Flow packets exported under the NetFlow version 9 protocol and handles flow record aggregation as well as representing both the record characteristics and geolocation results to the system user via a web page.

II. Network Flow

The concept of a Network Flow was patented by Kerr and Bruins on 28 May 1996 [3]. Initially, a flow was defined as a set of packets all destined for the same destination IP address and all originating from the same source address. Further identification of a unique flow included the requirement that all packets have the same destination port.

Since then, the concept of a network flow has been extended by Cisco to be defined as a unidirectional sequence of packets between two end hosts over a network [4]. Each packet in the sequence must display the same seven characteristics shown in Table I to be considered part of the same network flow. The direction of the flow is determined by the host that began the communication.

This flow data is generated by routing or switching devices such as those made by Cisco [5] and Juniper [6]. The generated data can then be transferred to other devices for analysis to help identify potential network faults and monitor resource use, typically for billing purposes. This data can further be analyzed to only display information pertaining to a particular network mask, a particular date or time, or overall resource use.

In order to transfer the recorded data that has been collected on observed flows to another device for processing, a raw flow exporting protocol is employed. The exporting can occur when the generating system concludes that a flow has expired or in set intervals. Currently, the most commonly used flow export protocols are those developed by Cisco which are NetFlow version 5 and NetFlow version 9 [7]. Additionally, a new protocol is under development by the Internet Engineering Task Force (IETF) called the IP Flow Information eXport (IPFIX) protocol [8]. Though still in development, implementations of the protocol are currently in use by network components such as the Barracuda NG Firewall[1].

[1]Only versions 5.2.3 and above are IPFIX compatible [9]

978-1-4799-3383-9/14/$31.00 ©2014 IEEE

TABLE I: Seven Characteristics of a Network Flow

Source IP Address
Destination IP Address
Source Port
Destination Port
Protocol
ToS Byte
Interface

A. NetFlow version 5

NetFlow version 5 is currently the most widely used protocol that is developed by Cisco for exporting raw flows from routing devices to the collector [10]. Packets exported under this protocol have a fixed size and a set number of fields. This fixes the number of characteristics relating to a particular raw flow that can be exported under this protocol.

Two disadvantages of using NetFlow version 5 over its successor variants are that it does not support IPv6 and the structure of exported packets is static [11]. IPv6 has become increasingly prevalent as services such those provided by Google are now accessible using IPv6 addressing. Flow records exported under the NetFlow version 5 protocol are limited to only exporting data in the defined fields of the record and cannot change during system runtime.

B. NetFlow version 9

NetFlow version 9 is a raw flow export protocol that dynamically structures the contents of the exported flow data records according to a previously exported template [7]. This allows collecting systems to process NetFlow version 9 data packets without knowing the format of the contained records prior to template lookup. The benefit of using this protocol is that it allows the characteristics exported to be changed without restarting either the exporter or collector systems. A result of this is that current NetFlow collectors and parser algorithms will not have to be recompiled to use a new packet structure when it becomes necessary to export a new characteristic from the export device. Another benefit of using a template based system is that companies can configure export devices to export flow records in a format that is optimal to their needs as well as modify them at a later stage without implementing a new protocol.

According to Cisco feature guides, NetFlow version 9 is independent of the implemented transport protocol being used to export the packets. Protocols used include UDP, TCP and SCTP as described in NetFlow configuration guide on exporting via SCTP [12].

Currently, NetFlow version 9 exports two types of records which are flow records and options records [7]. Flow records contain information regarding the generated flows and make up the majority of the records contained in the packet payload. Option records are used by the export devices to export additional information regarding the raw flows or the current configuration of the device. This can include the sampling frequency and the algorithm used to generate the flows as well as the number of flows observed.

C. IPFIX

IPFIX is a proposal to create a standardized protocol for the export of raw flow records from an export device to a collecting device. IPFIX is based on NetFlow version 9 with formal specifications being outlined in RFC 3917 [13]. IPFIX is well developed in terms of its specification and possible applications in industry.

The IPFIX protocol employs a dynamic packet structure consisting of data and template records. Though similar to NetFlow version 9, IPFIX extends the number of characteristics that can be exported from 128 to 239 [14]. As with NetFlow version 9, the collector requires that the template is received before any data formatted according to that template is received.

IPFIX is designed to be independent of the underlying transport protocol used which means it can be transmitted using IP layer protocols such as TCP, UDP, and SCTP. Though all three protocols are a viable choice for transmission, should the transmission path be susceptible to congestion, it is suggested in RFC 5101 [8] that SCTP be used rather than the other two protocols due to the protocol's congestion avoidance capabilities.

III. Geolocation

The Geolocation of IP addresses has become a useful resource to aid in targeted telemarketing as it allows for the generation of region specific advertisements [15]. Geolocation also has applications in cyber defense by aiding monitoring systems to identify the country or region in which an IP address of interest resides. Maxmind [16] and IP2Location [17] provide a range of databases that associate subnets with the country or region in which they reside. Online geolocation plugins are also available such as geoPlugin [18].

Investigations into applying geolocation to network flows has recently been discussed in a paper by Celeda et.al. [19]. This paper investigates the feasibility of applying geolocation to network flows for networks with high throughput. A comparison is done between applying the geolocation to recorded IP addresses at the exporter and the collector stages. This paper differs by focusing only on the collector side of the system with emphasis on the realtime visualization of geolocated network flows. Existing network flow visualizers such as WebView [20] and FlowViewer[2] exist and provide detailed visualization of a network using network flows. These are however not orientated towards realtime visualization which the tool described here attempts to achieve.

IV. System Design

The system we have developed was designed to achieve two defined goals. These goals were geolocation of the external endpoints of each flow and the generation of visual aids to assist in network visualization. In order to achieve these goals,

[2]http://sourceforge.net/projects/flowviewer/

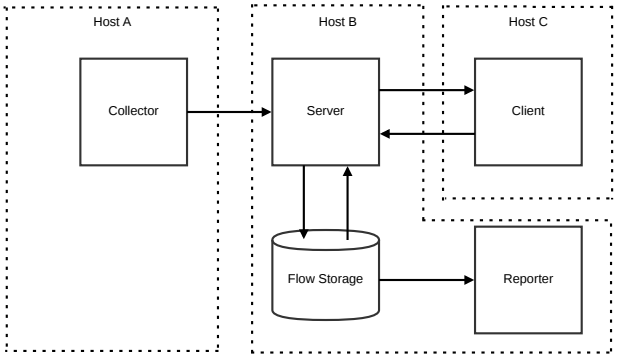

Fig. 1: System Overview

system components for geolocation, report generation, and visually representing recorded characteristics were needed.

To allow for interaction with the prototype system, the geolocation and some of the visualization components are displayed to the user using a web based interface. Communication with this interface is handled by a server component that can serve the web pages and maintain the connection between the served pages and the rest of the system. Furthermore, for the visualization components to function, the system also includes a database to act as a record storage component that can store the characteristics of the collected flow records.

The system is split into four main components which are necessary for the system to function. These components are; the Collector, the Server, the Client, and the Reporter. All of these components must be linked together to form the complete system which is is shown in Figure 1. In this figure arrows are used to represent the communication channels and the direction of data flow between the different components that make up the prototype system.

The four components are distributed into three different groups: hosts A, B, and C. The grouping indicates a direct dependency between the contained components while the communication between groups is done over TCP connections. By developing the system to communicate between the different groups over a network, the system can easily be fragmented with each group running on a different host.

By allowing the different components to communicate in this manor, it becomes possible to situate the different components in different networks so that the Server can become more accessible to a network segment that is separate from the segment in which the raw flows are being exported. This allows for improved security as the Clients do not need to access to the same network as the exporter to request pages from the server.

The Client should be designed to run in the host's browser and so is not required to be run on the same host as any other component of the system. Another import point to note about the Client is that the system should be capable of handling multiple instances of the Client simultaneously. To achieve this, the Client should be implemented as a web page which is returned by the Server to any requesting host.

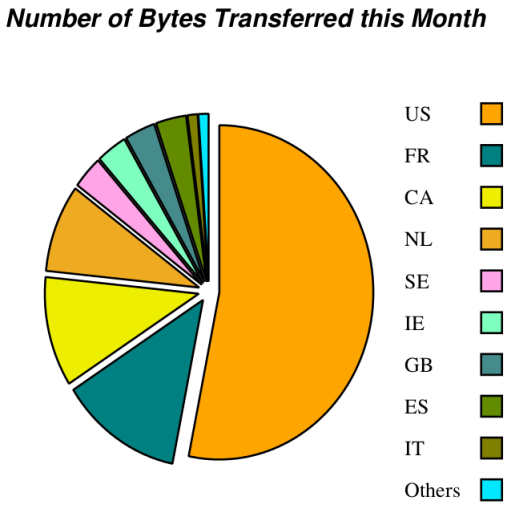

Fig. 2: System Overview

A. Constraints

The prototype system is only capable of recording flows relating to IPv4 address space and not the IPv6 address space. This constraining has been implemented to limit complexity during development and because of the lack of quality IPv6 golocation databases currently available.

The system implemented to perform this research is not currently capable of sending or receiving NetFlow options templates or NetFlow options records. The reason for this is that adding the additional functionality to the system does not effect the research performed.

V. System Implementation

Implementation of the system was done using python as it promotes rapid development and supports large collection of libraries that can easily be included for object serialisation and server implementation. Such libraries include Tornado[3], which provides asynchronous networking and a web framework, and pyGeoIP[4], an API for interfacing with geolocation databases.

Both the collector and Server components of the prototype system are implemented as multi threaded applications to allow for separate tasks to be completed as independent threads to prevent blocking. Unfortunately as the implementation language is Python, these multi threaded applications are unable to take advantage of a multicore platform.

The implementation of the reporter component of the system was done using the reportlab library. The library allows the programmer to create documents in Adobe's Portable Document Format (PDF) which can include both text and drawings [21]. The structure of the report is static and is generated procedurally by constructing a list of drawings that can be used to visualise the stored data. This list is then built into a PDF document which is returned by the reporter component. Figure 2 depicts an example of the total number of bytes transferred

[3]http://www.tornadoweb.org/en/stable/
[4]https://pypi.python.org/pypi/pygeoip

Fig. 3: Geolocated flow endpoint of IP: 146.231.123.92

Fig. 4: Google Maps lookup for IP: 146.231.123.92

for a specified month which is created as part of the report generation.

VI. Testing and Results

After implementation of the prototype system was complete, it was necessary to perform tests to insure that the geolocation component functioned correctly. Timing tests were also done for different system components as well as the system as a whole. From these timings we were able to determine the processing time by this system to convert a packet received under the NetFlow version 9 protocol into a flow event that could be sent to a web application for rendering.

A. Geolocation Results

In order to test the geolocation component of the prototype system, we used an external application to perform geolocation on a known IP address. The coordinates returned were rendered on a Google map to visibly display the address location. Simulated network traffic using these IP addresses was then replayed to the prototype system using an application called softflowd[5]. The rendered results were then recorded and compared with the Google map generated.

To generate the test traffic, two options were considered. The necessary packets could have been constructed entirely in the test bed or legitimate traffic could be collected and modified to suit the needs of the test bed. After considering the difficulties involved in either approach, it was decided that modifying legitimate traffic would be the optimal choice.

The legitimate traffic was generated using a simple Client/Server model which consists of a Client script and a

Fig. 5: Google Maps rendering of Physical[B] and Geolocated[A] location of IP 146.231.123.92.

Server script. The traffic generated from running this model was written to a pcap file using tcpdump[6]. The resulting pcap file was then read into a Python script which used the Scapy library to modify the packet headers. The modified data was then written into a new pcap file. To replay the modified traffic back onto the wire, tcpreplay was used which read the pcap file and reproduced the necessary packets.

The replayed traffic was processed by softflowd which in turn generated the raw flows and exported them to the implemented system. The system results were then rendered on the realtime map of a connected Client via the web browser. Comparing figures 4 and 3 it is shown that the results produced by the prototype system correlate with geolocation results produced by a different system. Geolocation tests were performed for other six different IP addresses and the rendered images for each show the same results.

Inaccuracies do however exist in the geolocation databases that the implemented system relies on [22]. The IP address geolocated in Figures 4 and 3 belong to Rhodes university which is not situated at the geolocated coordinates. The physical location of this institute is recorded using Google Maps as marker B in Figure 5. The distance between the actual location of the host and the geolocated position is 310.69 kilometers. Relative to a global scale, this inaccuracy is not large enough to consider geolocation unsuccessful but should warrant concern regarding geolocation accuracy. These inaccuracies arise because some subnets in the geolocation tables are incorrectly mapped and so is not a fault of the tool itself but rather in the geolocation database used.

B. Timing

The time taken for the implemented system to process a raw flow and produce a flow event is recorded to be approximately 0.05 seconds or 50 milliseconds. This is performed by recording the time when the system receives a raw network flow to the time the associated flow event is broadcast to the connected Clients. A subset of the recorded times are shown in Table II.

These results are measured from the time a packet is received by the Collector to when it is broadcast from the Server. As a result, it excludes the transmission time from the Server to the Client. It was decided not to perform timing tests

[5]http://code.google.com/p/softflowd/downloads/list

[6]http://www.tcpdump.org/release

TABLE II: Sample recorded results of time taken to process a network flow

Test No.	Time Packet Received	Time First Record Sent	Duration [s]
1	1381000626.1056400	1381000626.1600400	0.0543940
2	1381001156.9537300	1381001157.0111500	0.0574150
3	1381001211.8136500	1381001211.8485900	0.0349381
4	1381001301.8416500	1381001301.8949600	0.0533080
5	1381001356.3536600	1381001356.4073800	0.0537219

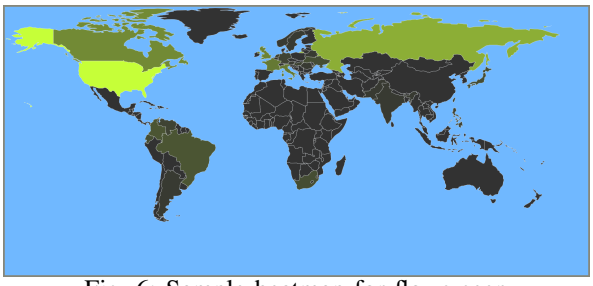

Fig. 6: Sample heatmap for flows seen

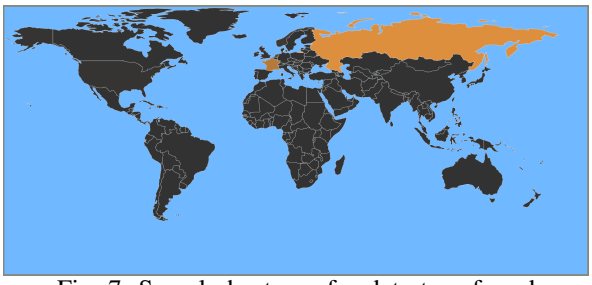

Fig. 7: Sample heatmap for data transferred

between the implemented Server and connected Clients as the duration is heavily dependant on the quality of the network path between the two components and considered out of scope for the prototype system

VII. System Demonstration

To demonstrate the functionality of the implemented system, a large file was downloaded using peer to peer file sharing system. The file downloaded was a distribution of the Linux Mint 15 operating system which is free to download[7] which is 959.4 MB in size.

The system was initialised and softflowd was used to monitor the download and generate the raw network flows. The file downloaded at an average of 18 MiB/s and a system report was generated. After the download had completed, the system was used to generate a series of heatmaps for the different flow characteristics.

From the generated maps, the comparison of two characteristics is of particular interest. The first characteristic is Figure 6 which is a heatmap where the colouring represents the number of flows seen to and from a specific country. The second characteristic depicts byte transfer where the heat indicates the volume of data seen from a specific country. The byte heatmap is depicted in Figure 7.

Figure 6 indicates the countries to which were connected to during the download. As expected, the heatmap shows results

[7]http://www.linuxmint.com/edition.php?id=132.

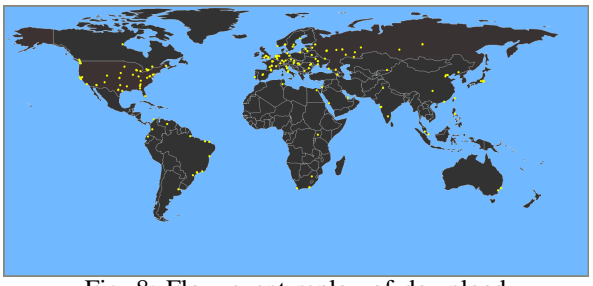

Fig. 8: Flow event replay of download

similar to the replay map shown in Figure 8 which shows the geolocated origins of these flows. Figure 7 describes a different result and shows countries Russia and France outweigh the rest of the world with regards to where the data was transferred from. This comparison shows how the system could be used to show the difference between flow count and flow volume. Similar comparisons can be done for packet count and the number of different IP addresses connected to in each country.

VIII. Conclusion

It is shown through conformance testing that the geolocation provided by the implemented system is suitable for geolocating network flows on a global scale. Inaccuracies do however exist but are due to inaccuracies in the geolocation database rather than a fault of the implemented system. This issue could be rectified by using a more accurate geolocation database but for the current implementation, the database used is still considered acceptable.

Considering the results collected from testing the prototype system, it can be concluded that network flows are a feasible source of data for traffic visualization and geolocation. Treating the exported characteristics associated with each flow as traffic summaries, analysis using network flows allows the user to quickly acquire a good overview of the condition of the associated network. A drawback to this approach is the loss of resolution and so the user will be unable to evaluate packet contents if using network flows.

IX. Future Work

Currently, this implementation of the system does not attempt to support IPv6 as it was not considered necessary for a proof of concept design. Should the system later be required to monitor and geolocate IPv6 addresses, then the system should be extended to do so.

As intended, protocols NetFLow version 9 and IPFIX are very similar so an extension to this system that should be considered is to implement support for IPFIX.

The reporting tools in the current implementation have been designed as a proof of concept and to show the type of information that the system can report on. This system component can be improved to include more thorough reporting tools, especially if the system is extended to accept a larger set of characteristics. The reporting functionality should also be tailored to suit the field that the system will be implemented in.

X. Acknowledgements

I would like to thank the NRF and Rhodes University for the financial support that allowed me to complete this research. I would also like to acknowledge the financial and technical support of Telkom, Tellabs, Stortech, Genband, Easttel, Bright Ideas 39 and THRIP through the Telkom Centre of Excellence in the Department of Computer Science at Rhodes University.

This research makes use of GeoLite data created by MaxMind.

References

[1] J. T. Morken, "Distributed NetFlow Processing Using the Map-Reduce Model," Computer and Information Science, Norwegian University of Science and Technology, June 2010, accessed on: 27 October 2013. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:352472/FULLTEXT01.pdf

[2] P. E. Proctor, *The Practical Intrusion Detection Handbook*, I. Winkler, Ed. Prentice Hall PTR, 2001, vol. 1, no. 1.

[3] Kerr and Bruins. (1996) Network flow switching and flow data export. Patent. [Online]. Available: http://www.lens.org/lens/patent/US7475156B2

[4] Cisco. (2012, October) NetFlow Services Solutions Guide . Cisco. Accessed on: 23 October 2013. [Online]. Available: http://www.cisco.com/en/US/docs/ios/solutions_docs/NetFlow/nfwhite.html

[5] ——, *Catalyst 6500/6000 Switches NetFlow Configuration and Troubleshooting*, Cisco, Jan 2012, document ID: 70974. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/70974-netflow-catalyst6500.html

[6] I. Juniper Networks. (2013) Junos OS Routing Protocols Overview. Accessed on 21 May 2013. [Online]. Available: http://www.juniper.net/techpubs/en_US/junos13.1/information-products/pathway-pages/config-guide-routing/config-guide-routing-overview.pdf

[7] Cisco. (2011) Cisco IOS NetFlow Version 9 Flow-Record Format. Accessed on 16 May 2013. [Online]. Available: http://www.cisco.com/en/US/technologies/tk648/tk362/technologies_white_paper09186a00800a3db9.pdf

[8] Claise, Cisco Systems, Trammell, and Zurich, *Specification of the IP Flow Information eXport (IPFIX) Protocol for the Exchange of Flow Information*, RFC, Network Working Group Std., Rev. rfc5101bis-06, Feb 2013. [Online]. Available: http://tools.ietf.org/html/rfc7011

[9] Barracuda. (2013, May) How to Configure Audit & Reporting With IPFIX. TechLibrary. Barracuda. Accessed on 27 October 2013. [Online]. Available: http://techlib.barracuda.com/pages/viewpage.action?pageId=6979841

[10] C. Lee, H. Kim, H. Jeong, and Y. Won, "Analysis of SIP Traffic Behavior with NetFlow - based Statistical Information," June 2010.

[11] Cisco Systems, *Catalyst 4500 Series Switch Cisco IOS Software Configuration Guide*, 12th ed., Cisco, Cisco Systems, Inc. 170 West Tasman Drive San Jose, CA 95134-1706 USA, 2004, accessed on: September 2013. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/54sg/configuration/guide/config.pdf

[12] Cisco. (2012) NetFlow Reliable Export With SCTP. Accessed on 14 May 2013. [Online]. Available: http://www.cisco.com/c/en/us/td/docs/ios/12_4t/12_4t4/nfhtsctp.html

[13] T. Zseby, J. Quittek, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," Internet Engineering Task Force(IETF), Tech. Rep., 2004. [Online]. Available: http://tools.ietf.org/html/rfc3917

[14] J. Quittek, P. Aitken, J. Meyer, B. Claise, and S. Bryant, "Information model for ip flow information export," IETF, Tech. Rep., 2008, accessed on: March 2013. [Online]. Available: http://tools.ietf.org/html/rfc5102

[15] G. Cliquet, *Geomarketing. Methods and Strategies in Spatial Thinking*, G. Cliquet, Ed. ISTE Ltd, 2006, vol. 1.

[16] MaxMind. Maxmind. MaxMind Inc. Accessed on 12 May 2013. [Online]. Available: http://www.maxmind.com/en/home

[17] IP2Location. IP Address Geolocation to Identify Website Visitors Geographical Location. Online. IP2Location. Accessed on: 15th October 2013. [Online]. Available: http://www.ip2location.com/

[18] geoPlugin. plugin to geo-targeting and unleash your site's potential. Online. geoPlugin. Accessed on 5 March 2014. [Online]. Available: http://www.geoplugin.com/start

[19] P. Celeda, P. Velan, and M. Rabek, "Large-Scale Geolocation for NetFlow," *IFIP/IEEE International Symposium on Integrated Network Management*, no. 13, pp. 1015–1020, May 2013, accessed on 1 July 2014.

[20] C. Weinhold. (2013) Webview Netflow Reporter. Online. CDW. Accessed on 1 July 2014. [Online]. Available: http://wvnetflow.sourceforge.net/#overview

[21] ReportLab, "Reportlab pdf library user guide," ReportLab, ReportLab PDF Library User Guide ReportLab Version 2.7 Document generated on 2013/05/07 20:18:53 Thornton House Thornton Road Wimbledon London SW19 4NG, UK, Users Guide 2.7, May 2013. [Online]. Available: http://www.reportlab.com/docs/reportlab-userguide.pdf

[22] I. Poese, S. Uhlig, and M. Ali Kaafar. IP Geolocation Databases: Unreliable? CCR. Accessed on 5 May 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=1971171