Deep Packet Inspection - Fear of the Unknown

Ryan Goss
Institute for ICT Advancement & School of ICT
Nelson Mandela Metropolitan University
Port Elizabeth, South Africa
Email: ryan@goss.co.za

Reinhardt Botha
Institute for ICT Advancement & School of ICT
Nelson Mandela Metropolitan University
Port Elizabeth, South Africa
Email: reinhardta.botha@nmmu.ac.za

***Abstract*—Enterprise and service provider customers develop, maintain and operate network infrastructure in order to support the applications required to perform their day to day tasks. These applications have certain requirements and expectations from the infrastructure, including access to public networks, and thus rely on quality of service (QoS) controls to manage network traffic. QoS controls are used to ensure non-critical applications do not hamper the operation of critical ones, all the while providing fair access to all legitimate applications. QoS systems are increasingly being used as firewalls, filtering bad traffic and allowing good traffic to traverse the network without delay.**

This paper investigates the effectiveness of protocol matching within current QoS classifiers and shows that even with the most up to date classifiers, "unknown" or unidentified traffic is still prevalent on a network; a serious concern for IT network administrators. This "unknown traffic could consist of viruses, attempted exploits and other un-authorized connectivity from outside sources.

I. Introduction

Communication has, since the inception of the personal computer system, become a vital lifeline in modern society for businesses and individuals alike. The increase in need for communication has resulted in the development of faster, more connected computer networks over a vast geographical area [1]. The fastest way to improve global reach of communication is to harness the power of broader reaching public networks, such as the Internet, for day to day communications, for both personal and business use. Connecting to public networks, specifically the Internet, introduces a variety of potential attacks and unwanted traffic to once closed, protected computer systems. These unwanted traffic flows may slow throughput, degrade communication performance and potentially expose these computer systems to a wide variety of direct and indirect attacks. It is therefore necessary to separate "Good traffic" from "Bad traffic" to optimize performance of communication networks and mitigate the risk imposed on these once closed computer systems. Internet firewalls and intrusion detection systems have thus subsequently become a vital component of these global networks [2].

These devices, which were initially designed solely with "best effort" packet switching on a first-come-first-served basis in mind, have been called upon to provide different qualities of service to manage traffic [3]. The enhanced traffic management requirements include admission control, resource reservation, per-flow queueing and fair scheduling. These features require the router to distinguish packets belonging to different flows, marking them as related in order to effectively route and manage each individual connection. The traffic flows have previously been specified by various predefined rules, applied to all incoming packets through a router's network interface. This collection of rules are collectively known as a *classifier* [3].

The rules of a traditional classifier follow the convention that rules closer to the top of the list take priority [3]. The rules operate on a pass-through basis whereby the rule set is traversed until a match has been made, at which time a mark is set and applied to the flow. These marks have traditionally been set based on simplistic packet header information [2] and limited by syntax to a simple address/mask or operator/number(s) specification such as source or destination ports [3].

Past firewall matching capabilities were shown to focus on more simplistic traffic matching mechanisms, such as port and protocol comparisons [4]. Recent advances in network applications make use of application layer communication protocols, designed to circumvent simple protocol matching such as port and IP address information found in packet headers [5], [6]. Peer-to-peer applications such as Skype, Bittorrent and the GNUtella network applications do not operate on standard or fixed ports, protocols or IP addresses [7], but rather randomly select ports on which to send and receive data [6], [8]. These applications account for up to 50%-70% of Internet traffic [7] and are designed to pass through firewalls and traffic shaping mechanisms [8] imposed by network administrators. These rules are put in place in order to ensure the organizational policies governing network usage are strictly adhered to and that the network is optimized for its primary use. Enhancements in firewall filtering techniques have adapted to look into the underlying packet payload of network traffic to classify it, rather than solely the packet headers [9].

It is important to classify network traffic in order to assist network administrators in managing data flows on their IP networks [7], [10]. By providing network administrators the ability to flag traffic flows with a particular firewall mark, they are able to block, restrict flow (traffic-shape) and possibly redirect certain flows based on a predefined ruleset. This process increases the overall security of the network as unclassified traffic will not go unnoticed as is the case in many network

978-1-4244-5494-5/10/$26.00 ©2010 IEEE

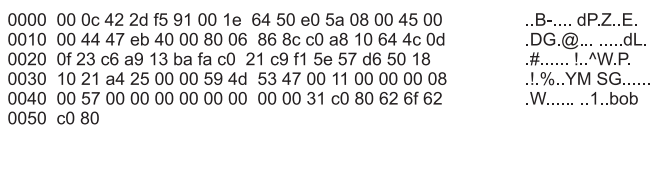

Fig. 1. Wireshark Snapshot of the Yahoo! Messenger Protocol

implementations.

Projects such as the L7-Filter project [12] and the Open Deep Packet Inspection Engine project [13] attempts to address this problem through deep packet inspection techniques. By using regular expressions, deep packet inspection systems attempt to match patterns or *signatures* in the packet payload [7]. Commercial applications also exist with their own classification engines, designed using deep packet inspection techniques. Some examples of these systems include the Cisco nBAR application [14], Bluecoat appliances [15] and Checkpoint firewalls [16].

Many systems focus on the use of regular expressions to match strings in the underlying packet payload [11]. The regular expressions used in these systems usually originate from the manufacturer of the system, or network administrators who create their own. The requirement for manual creation of a classifier to match each protocol has at least one major drawback - if no classifiers have been released or created for a particular protocol, that particular protocol will traverse the network as "unknown" traffic. This design promotes scalability issues for the system due to the unavailability of sufficient pattern signatures [7].

Each regular expression describes a set of strings without enumerating them explicitly [11]. Consider for example the following regular expression which matches the popular Yahoo! Messenger traffic flow [11]:

```
^ (ymsg|ypns|yhoo).?.?.?.?.?.?.?[lwt].*
\xc0 \x80
```

The QoS system would attempt to match the data payload contained within the first few packets of each connection to the regular expression. If no match is found, it tries the next regular expression until either a match is found, or the connection is marked as "unknown".

An example of a packet payload containing Yahoo! information was captured using the Wireshark application and is shown in Figure 1. In amongst the various other characters, this packet's payload included a string of characters which would cause a match if compared to the Yahoo! messenger protocol's classifier.

A hexidecimal representation of the packet's payload is shown on the left hand side, with the ASCII representation on the right. As can be seen, the characters "YMSG" exist, followed by 7 other characters and then the "W" character. Finally, a few characters later the characters c0 and 80 are found, terminating the string.

The requirement for manual creation of regular expressions leads to a potential delay in the ability to match certain protocols which may be essential to network administrators, such as new virus attacks or worms. Any traffic not matched by existing regular expressions are generally marked with the "unknown" classifier. However, the unknown classifier does not help the administrator or network security engineer establish the legitimacy of the underlying traffic flow. This provides potential for the administrators to either accidentally block good traffic, or unblock bad traffic.

The question thus posed by this paper relates directly to the operation of these systems: Just how much of the traffic flowing on a particular network is able to be matched by existing, fully updated classification systems?

In order to answer this question, the following experiment was setup to establish the status quo of classifiers currently available - from both commercial and open source offerings.

II. Experiment

In order for a deep packet inspection engine to be effective in matching traffic, both directions of traffic flow need to be seen by the classifier. The traffic generated by the source and the replies generated by the destination thus need to be passed through the classifier.

This experiment was setup in such a way that a particular set of traffic was recorded, so that it could be replayed to multiple classification engines and thus ascertain the effectiveness of each in a controlled test environment. As the traffic was being recorded and replayed, the experiment would not be able to replay replies from hosts which were not part of the local network, due to the potential inability to record such traffic. For this reason, the traffic recorded and replayed was that which was sent and received on the local network segment.

A. Data set

The data set used for this experiment was captured on the local area network of an information technology company, using Wireshark [17] on a Linux based host machine. Permission was obtained from the operations manager of the organization to record traffic flows traversing their network during a 1 hour period of a normal business day. Users were aware of the experiment and that the information generated for the course of the day was subject to being recorded, in line with Section 86 of the South African Electronic Communications and Transactions Act 25 of 2002. The data set comprised of traffic flows generated by users during the day for the purposes of conducting their work and included HTTP, SSH, FTP, Instant Messenger and Email traffic flow information within the local network. Further to this, a small amount of Peer to Peer based traffic was also recorded, which consisted of Skype [18] and Bittorrent [19] protocols. Filters configured on the Wireshark application ensured that only local traffic was recorded, ensuring that when the traffic was replayed in the offline test environment, both directions of the traffic flow would be seen by the classifier.

978-1-4244-5494-5/10/$26.00 ©2010 IEEE

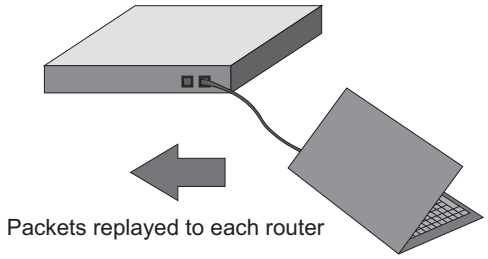

Fig. 2. Network Topology Design

The data was recorded from 13:30 until 14:30 on a Monday, with the time difference between packets also captured so that during the replay of the traffic on the test network, the exact packet delays could also be simulated, rather than replaying the packets in a seemingly uncontrolled manner.

B. Test Environment

A simplistic network topology, illustrated in Figure 2, was setup in order to insert the routers whose classifiers would be tested and subject them to the recorded data set. All tests were performed in a totally isolated network setup, away from any external traffic influences.

The authors decided to gauge the effectiveness of both the open source and commercial deep packet inspection offerings. The routers were thus selected based on their availability at time of testing. Both routers were subjected to the same test data. This was done with two objectives. First, to ascertain which percentage of the traffic was identified by each. Second, to compare results and determine on average how much traffic were "unknown", i.e. it could not be classified.

Based on these requirements, the following two deep packet inspection engines were selected for testing:

- Cisco[1] Network Based Application Recognition (nBAR) - A commercial application developed by Cisco with the intent on matching Real-time Transport Protocol (RTP) and other applications using deep packet inspection techniques.
- Linux based router running L7 Filtering - Mikrotik RouterOS[2] offers support for using regular expressions to match connections using the underlying packet payload with an IP network traffic flow, using the protocols created and managed by the Open Source L7-filter project.

The Cisco router used was a 2621 series, running version 12.3 of Internetwork Operating System (IOS) and NBAR application produced by Cisco Systems. The Mikrotik router was a Routerboard 433, running RouterOS version 3.33 with layer 7 filter support. Each router was in turn inserted into the test network shown described in Figure 2, with its primary ethernet interface directly connected to a laptop running Windows 7. This laptop stored the information recorded by the Linux host machine and had the application installed to replay the traffic to the test routers.

[1]http://www.cisco.com/
[2]http://www.mikrotik.com/

C. Test Execution

In order to ensure the best possible chance for maximum traffic identification, each system was updated with the latest version of their operating system and protocol information databases (signatures). These signatures were loaded into the system's forwarding chain, allowing them to count the bytes and mark connections associated with the various traffic flows. Although the results could provide information as to the accuracy of these systems, including false positives and false negatives for each protocol, the focus of this paper is merely to ascertain traffic matched versus the amount of unidentified traffic, marked "unknown" by the system. It should further be noted that the tests conducted during this experiment were performed purely with deep packet inspection filters. Standard port-based filters were purposefully excluded in order to indicate the accuracy of the router's deep packet inspection engine and classifiers.

Each router was in turn placed into the test network and had its primary IP address for the interface connected to the laptop configured on the same range as the subject network was running. An application called PrePlay [20] running on the Windows 7 laptop was used to read the file produced by the Wireshark application and replay this traffic to each of the routers. After the data set had been transmitted (approximately 1 hour for each), the byte and packet counters on each router were recorded and documented in Table I.

III. Results

The results observed after replaying the 27,502 packets (totalling 16,088,348 bytes) of the sample data set on each router independently yielded the following results:

TABLE I
RESULTS OF EXPERIMENT

	Known	Unknown	Classifiers
Cisco nBAR	87.44%	12.56%	71
Mikrotik	69.56%	30.44%	105
Average	78.5%	21.5%	

The results recovered from this experiment indicate that the open-source application, although having significantly more classifiers than the commercial application, failed to match as much of the traffic as the Cisco nBAR product. Even so, the results were not too far apart, providing enough information to formulate a fair set of findings. The averages for the two application's results were also calculated and are shown at the bottom of the table.

IV. Discussion

The results produced by this experiment indicate that although a significant amount of research has been conducted into the improvement of deep packet inspection engines, there

is still a lot more work to be done to achieve perfection. This paper has shown that a significant portion of all traffic recorded on the network was unable to be identified by popular deep packet inspection engines.

Comparing the results between the two systems tested, it can clearly be seen that the open source offerings lacked in identification compared with the commercial product. The open source system, although boasting more than 34 additional classifiers than the Cisco product, resulted in almost 18% less identifications. Whilst commercial systems may have a lot more time and resources invested with their development, open source systems are generally able to produce significant results due to the overwhelming support of the community of developers. The open source systems inability to identify as much as the Cisco product may thus be due to the classifiers themselves being too specific. Alternatively, the implementation of the deep packet inspection techniques used by the system may not be as efficient or optimized as the various commercial offerings.

Although the commercial product in this test appeared to outperform the open source one, it can be seen that there is a significant amount of "unknown" traffic which neither system classified. This result appears to be directly due to the effectiveness of each of the classifiers of the systems. The traffic that was identified by each system may also not be correct in that the classifier may over generalize in order to make a match, rather than very specific which would limit the results significantly. These classifiers may result in traffic being skipped, instead of it being matched by the system. It is therefore possible that the unknown traffic may wrongfully contain traffic which should be known to at least one of the classifiers of the system, or that certain known traffic should in fact have been marked as unknown.

As more and more applications are developed to elude standard QoS techniques and deep packet inspection classfiers, the amount of unknown traffic traversing networks will increase. The unavailability of classifiers for a new protocol prevents the network administrator from managing the traffic flow until one becomes available. In contrast, perhaps there was a classifier designed with the intention of matching the specific traffic, however problems within the regular expression caused it to fail to match the protocol. Human error in the creation and maintenance of classifiers thus plays a significant part in the accuracy and effectiveness of the system.

Another method of eluding these systems includes applications which opt to encrypt the underlying protocol information contained within the data packet. Upon inspection of the data field of such packets, the classifiers will generally not match any of the strings due to them being encrypted. This alone forms a large area for future researchers to investigate when ascertaining the effectiveness of deep packet inspection systems.

This paper serves as a basis for future research in various areas of deep packet inspection. The correctness of the known and the unknown traffic needs to be addressed - just how accurate are the classifiers when marking traffic? How could the accuracy of these systems be increased without incurring any additional penalties such as slower throughput due to processing time. The development of techniques to handle unknown traffic in a controlled manner is also an area worth investigating. Simply allowing this data to flow on the network in a seemingly uncontrolled manner may not be enough for certain network administrators. It may be necessary to implement controls whereby this data can be managed and prioritized in a structured, controlled manner.

There are numerous ways to validate the findings of this paper in more detail. This includes the broadening of the base of the experiment, by including more deep packet inspection products into the experiment and by comparing more than one set of sample data. This paper has shown that popular deep packet inspection engines are unable to identify a significant portion of all traffic recorded on the network.

V. Conclusion

The results produced by this experiment indicate that there is a clear requirement for improving existing traffic classification systems. Application protocols are adapting to the signatures deployed by manufacturers of firewall applications at a rate that the manufacturers are unable to keep up with. This results in an increasing percentage of "unknown" traffic on a corporate network, lowering the overall security and control of the data network.

Based on the average results obtained through the experiment described in this paper, almost a quarter of all network traffic observed on the subject network was classified as "unknown" by the firewall, leaving a significant amount of unmanageable traffic traversing the network with no foreseeable way of managing it. This provides a cause for concern for network administrators, who attempt to manage their networks in a secure and efficient manner.

References

[1] Curtis, N., & Taylor, P. (2008). Network architecture. The Independent Institute of Education.

[2] Moscola, J., Lockwood, J., Loui, R. P., & Pachos, M. (2003, April). Implementation of a content-scanning module for an internet firewall. In Proceedings of the 11th Annual IEEE Symposium on Field Programmable Custom Computing Machines (p. 31). IEEE Computer Society.

[3] Gupta, P., & McKeown, N. (2001, March). Algorithms for packet classification. IEEE Network, 15 (2), 24-32

[4] Zhang, J., Qian, Z., Shou, G. & Hu, Y. (2009). An Automated Online Traffic Flow Classification Scheme. In Proceedings of the 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing.

[5] Renals, P., & Jacoby, G. A. (2009, January). Blocking skype through deep packet inspection. In Proceedings of the 42nd Hawaii International Conference on System Sciences. Waikoloa, Big Island, Hawaii.

[6] Kim, H., Claffy, K., Fomenkov, M., Barman, D., Faloutsos, M., & Lee, K. (2008, December). Internet traffic classification demystified: Myths, caveats, and the best practices. In CoNEXT Proceedings of the 2008 ACM CoNEXT Conference (pp. 1 - 12). Madrid, Spain: ACM.

[7] Xie, X., Yang, B., Chen, Y., Wang, L., & Chen, Z. (2009). Network traffic classification based on error-correcting output codes and nn ensemble. In Proceedings of the 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery (pp. 475 - 479). Tianjin, China.

[8] Baset, S. A., & Schulzrinne, H. G. (2006, April). An analysis of the skype peer-to-peer internet telephony protocol. In Proceedings of the 25th Conference on Computer Communications (IEEE Infocom 2006) (p. 1-11). Barcelona, Spain.

978-1-4244-5494-5/10/$26.00 ©2010 IEEE

[9] Dharmapurikar, S., Krishnamurthy, P., Sproull, T., & Lockwood, J. (2004, January). Deep packet inspection using parallel bloom filters. IEEE Micro, 24 (1), 52 - 61.
[10] Lakhina, A., Crovella, M., & Diot, C. (2004). Characterization of network-wide anomalies in traffic flows. In IMC Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (pp. 201 - 206). Taormina, Sicily, Italy.
[11] Yu, F., Chen, Z., Diao, Y., Lakshman, T., & Katz, R. H. (2006). Fast and memory-efficient regular expression matching for deep packet inspection. In Proceedings of the 2006 ACM/IEEE symposium on Architecture for Networking and Communications Systems (pp. 93 - 102). San Jose, California, USA: ACM Press.
[12] The L7-Filter project *http://l7-filter.sourceforge.net/*
[13] The Open Deep Packet Inspection Project *http://www.opendpi.org/*
[14] Cisco nBar Documentation *http://www.cisco.com/*
[15] Bluecoat Proxy and Firewalls *http://www.bluecoat.com/*
[16] Checkpoint firewalls *http://www.checkpoint.com/*
[17] Wireshark Network Traffic Monitor *http://www.wireshark.org/*
[18] Skype *http://www.skype.com/*
[19] Bittorrent Homepage *http://www.bittorrent.com/*
[20] PrePlay Homepage *http://www.secgeeks.com/*

978-1-4244-5494-5/10/$26.00 ©2010 IEEE