Sneaking Past the Firewall: Quantifying the Unexpected Traffic on Major TCP and UDP Ports

Shane Alcock
University of Waikato
Hamilton, New Zealand
salcock@waikato.ac.nz

Jean-Pierre Möller
University of Waikato
Hamilton, New Zealand
jp.moller78@gmail.com

Richard Nelson
University of Waikato
Hamilton, New Zealand
richardn@waikato.ac.nz

ABSTRACT

This study aims to identify and quantify applications that are making use of port numbers that are typically associated with other major Internet applications (i.e. port 53, 80, 123, 443, 8000 and 8080) to bypass port-based traffic controls such as firewalls. We use lightweight packet inspection to examine each flow observed using these ports on our campus network over the course of a week in September 2015 and identify applications that are producing network traffic that does not match the expected application for each port. We find that there are numerous programs that co-opt the port numbers of major Internet applications on our campus, many of which are Chinese in origin and are not recognized by existing traffic classification tools. As a result of our investigation, new rules for identifying over 20 new applications have been made available to the research community.

Keywords

traffic classification; firewalls; application protocols

1. INTRODUCTION

Practical network operations are still heavily dependent on port numbers from transport layer headers (i.e. TCP and UDP) for traffic control purposes [6]; firewall rules being one of the more obvious examples. Port-based controls are popular because they are easy, and therefore cheap, to implement in hardware and because they are simple for operators to comprehend. The problem with port-based approaches is that port numbers cannot reliably identify all applications. Peer-to-peer (P2P) applications, in particular, have long been using random port numbers to avoid having their traffic dropped or rate-limited by network operators [11]. In response, standard operating practice has moved towards a 'default deny' approach to firewalling, particularly within corporate or campus networks, where only traffic on ports associated with essential services is allowed to traverse protected segments of the network.

However, this ignores another problem: what if other applications start using the ports that are typically associated with the essential services? Previous research has already identified that not all TCP port 80 traffic is HTTP [5] [8] [10] [15], although in each study the 'unexpected' traffic on TCP port 80 was not analyzed in depth or associated with particular applications. Other ports used by essential services have been ignored in past research, such as TCP port 443 (HTTPS), UDP port 53 (DNS) and the alternative HTTP ports (TCP port 8080 and 8000). As a result, we know that these ports may be sometimes used by other applications but little about which applications are doing so and the amount of traffic that these applications are 'sneaking' past port-based traffic controls.

In this paper, we investigate and catalogue unexpected traffic on the set of ports that are most likely to be open on many, if not all, firewalls: port 53, 80, 443, 8000, 8080 and 123. Our aim is to not just demonstrate that unexpected traffic exists on these ports (as this has already been established in previous literature), but to also investigate further and try to correctly identify the applications responsible for as much of the unexpected traffic as possible.

Using a week-long packet capture from our campus network and the libprotoident [1] traffic classification library, we have successfully identified many of the applications that were using these port numbers and can therefore bypass a typical firewall configuration undetected. We also quantify the effect of these applications in terms of the number of flows and amount of bytes that each application was contributing. Many of the applications that we identified originate from China and, as far as we are aware, were not able to be recognized by existing open-source traffic classifiers prior to this

IMC '16, November 14–16, Santa Monica, CA, USA.

DOI: http://dx.doi.org/10.1145/2987443.2987447

research being conducted. Rules for 22 new application protocols have been added to libprotoident and made freely available through the latest software release.

Because our analysis is from a single campus network with a high proportion of Chinese and Taiwanese overseas students (8.4% of students are from China / Taiwan), we recognize that it would not be appropriate to generalize our observations to the Internet as a whole. Rather, we present this study as an single data point that can be compared against by people with access to datasets from other networks with different demographics. To aid in the continuation of study in this area, the new protocol matching rules that we developed have been included in the latest libprotoident release and are therefore freely available to other researchers.

2. METHODOLOGY

2.1 Packet Traces

The traceset used in this analysis consists of seven days and seven hours of packet capture taken in September 2015 at the University of Waikato, during a teaching semester at the University. During this period, 8.4% of enrolled students (1,048 out of 12,502 enrolments) were international students from either China or Taiwan; the relevance of this statistic will become apparent when we present our results in Section 3.

The packets were captured using an Endace Probe that was installed at the edge of the campus network, outside the campus firewall. In this location, the Probe was able to observe all traffic entering and exiting the University. The packets captured by the Probe were forwarded to an instance of our custom WDCap trace capture software [17] that was running in a virtual machine on the probe. WDCap was configured to anonymize the local IP address (i.e. the University address) using the standard prefix-preserving CryptoPAn method [9] and to truncate each packet to contain no more than four bytes of post-transport header payload. Both of these modifications were to preserve the privacy of the network users. The remote IP address was left unmodified (with permission from the University); this was primarily to aid us in identifying the source of new applications encountered during our analysis.

2.2 Libprotoident

Libprotoident [1] is a software library for lightweight traffic classification that can identify the application protocol being used by network flows based on the contents of the first four bytes of payload sent in each direction, the size of the first packet sent in each direction and the port numbers used by each endpoint. Unlike some classification techniques, libprotoident assumes the IP addresses for each flow are anonymized so the addresses are not used for classification. Despite using such limited information, libprotoident has compared favorably with other open-source DPI-based classifiers in terms of accuracy [2] [4].

The analysis presented in this paper was conducted using the `lpi_protoident` tool that is included with the libprotoident library. This tool prints a single line of output for each bidirectional flow found in the input traffic dataset containing the flow 5-tuple, the time that the flow began, the number of bytes sent in each direction and the application protocol that libprotoident has assigned to that flow. TCP flows where a complete handshake was not observed are ignored, as we cannot be certain that the first payload-bearing packets for those flows will have been observed. For UDP, we already have to accept that the first payload-bearing packets may be missing, even if the flow begins and ends within the time period covered by our dataset, so all observed flows are reported. Flows are expired after either a period of inactivity (2 minutes for UDP flows and unestablished TCP flows, 2 hours for established TCP flows) or the observation of a connection termination signal (e.g. a TCP RST or a FIN in both directions).

To prevent scans, backscatter and other unsolicited traffic from affecting our results, we ignored all flows where both endpoints did not participate in the flow. For TCP flows, the handshake must have been completed successfully and at least one endpoint must have sent a packet bearing payload. For UDP flows, only flows where both endpoints had sent a payload-bearing packet have been included in the analysis.

Our initial attempt at analysis with the most recent libprotoident release resulted in a significant quantity of unidentified traffic. As a result, much of the research effort in this study was in developing new libprotoident rules to reduce the amount of unidentified traffic in our dataset as much as possible. As a result, we have been able to add rules to libprotoident for 22 new applications, as well as improve the quality of the rules for a further 10 applications that were already supported. Most of the new applications are Chinese in origin and include Kankan, Kakao, Weibo, Kugou and Xiami.

Space limitations mean that we are unable to fully describe the changes to libprotoident here; instead, we have created a companion webpage [1] for interested readers with more detail on the new applications that we have identified as a result of this work and the rules that we have written to match them. The new and improved protocol rules have also been made available publicly in the 2.0.8 release of libprotoident, which is open-sourced under the GPL.

3. RESULTS

Table 1 shows the amount of expected and unexpected traffic in our dataset for each of the ports that we investigated. We consider traffic on a port to be 'unexpected' if the application protocol does not match the protocol that one would expect to see on that port. For each port number, we examined both TCP and UDP

[1] https://secure.wand.net.nz/content/sneaking-past-firewall-paper-addendum

Port	Protocol	Expected Bytes	Expected Flows	Unexpected Bytes	Unexpected Flows
TCP 53	DNS	1.4 G (74.6 %)	1.8 M (93.0 %)	463 M (25.4 %)	137 K (7.0 %)
TCP 80	HTTP	11 T (99.8 %)	59 M (99.3 %)	26 G (0.2 %)	406 K (0.7 %)
TCP 123	NTP	None (0.00 %)	0 (0.00%)	293 K (100 %)	236 (100 %)
TCP 443	HTTPS	12 T (99.3 %)	64 M (99.3 %)	89 G (0.7 %)	421 K (0.7 %)
TCP 8000	HTTP	4.2 G (98.9 %)	18 K (61.5 %)	45 M (1.1 %)	11 K (38.5 %)
TCP 8080	HTTP	24 G (32.6 %)	193 K (63.0 %)	50 G (67.4 %)	113 K (37.0 %)
UDP 53	DNS	127 G (95.3 %)	364 G (99.97 %)	6.2 G (4.7 %)	91 K (0.03 %)
UDP 80	HTTP	None (0.00 %)	0 (0.00%)	7 G (100 %)	90 K (100 %)
UDP 123	NTP	16 G (99.99 %)	123 M (99.99 %)	274 K (0.01 %)	609 (0.01 %)
UDP 443	QUIC	1.4 T (99.8 %)	3.4 M (97.3 %)	2.2 G (0.2 %)	93 K (2.7 %)
UDP 8000	HTTP	None (0.00 %)	0 (0.00%)	13 G (100 %)	167 K (100 %)
UDP 8080	HTTP	None (0.00 %)	0 (0.00%)	38 G (100 %)	30 K (100 %)

Table 1: Expected and unexpected traffic volumes on each port analyzed for this study.

traffic as the IANA port number registry often assigns both the TCP and UDP port numbers to the same application (for example, UDP port 80 is assigned to HTTP), so it is possible that an operator might open these ports for both transport protocols.

Table 1 includes counts for both traffic volumes (as bytes) and unique flows. We designated QUIC as the expected protocol for UDP port 443, even though the IANA registry has UDP port 443 assigned as HTTPS, because it was clear from our initial analysis that QUIC was the primary user of this port. The volume of QUIC traffic that we observed on our campus network was much higher than we had anticipated (1.4 TB), almost all of which was coming from Google Global Cache nodes. This demonstrates that Google is now using QUIC to distribute a large proportion of content, which we believe will have a major impact on how Google services, especially YouTube, are measured and monitored.

Only in five out of twelve cases did the expected protocol accounts for more than 95% of the flows and bytes observed for a port number. Even in these cases, a closer examination of the absolute quantities reveals that there was still a significant amount of unexpected traffic present on some ports. For instance, there was 89 GB of non-HTTPS traffic on TCP port 443 and a further 26 GB of non-HTTP traffic on TCP port 80. The remaining seven ports have been more clearly co-opted by other applications. On TCP port 8080, the amount of HTTP traffic was outweighed by non-HTTP traffic by a ratio of 2:1 and there was 38 GB of traffic observed on UDP port 8080, none of which matched the protocol assigned to that port by IANA.

Tables 2 through to 11 show the most prominent unexpected applications observed on each of the analyzed ports. We have omitted the tables for TCP and UDP port 123 as these ports had so little unexpected traffic that they did not warrant further analysis. In the tables, 'Unknown' represents flows that libprotoident was unable to assign to a particular protocol and 'Other' refers to flows that matched a protocol that neither contributed a sufficient number of bytes or flows to deserve a separate row. The number of unique applications represented by the 'Other' category are included in parentheses next to the label.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
Youku	93.3	0.71	776	12
Unknown	4.48	88.9	345	45
OpenVPN	2.20	10.3	188	31
Other (2)	0.02	0.09	-	-

Table 2: Non-DNS Applications observed using TCP port 53 in the Waikato traces.

We have also included the number of unique local and remote IP addresses (as they appeared in the IP headers of the packets in our traces) that were observed using each application. These numbers are included to demonstrate that much of the behavior that we are reporting is not necessarily limited to just one or two network users or servers. However, because many of the student subnets on our campus use DHCP to lease IP addresses to users and a single user could therefore lease multiple addresses over the duration of our dataset, the local IP numbers should be seen as only an approximation of the number of users for each application.

3.1 TCP Ports

Youku, a Chinese video hosting and streaming service similar to YouTube, was responsible for 93.3% of the unexpected traffic on TCP port 53, as shown in Table 2. The number of Youku flows observed was small: less than 1% of unexpected flows. Most of the unexpected flows were categorized as Unknown by libprotoident; manual analysis of these flows suggested that they are most likely DNS requests that did not receive a response. Because libprotoident cannot be certain that the traffic was DNS without seeing the identification field of the response packet, the flows were reported as Unknown. The only other protocol to feature prominently on TCP port 53 was OpenVPN, which was responsible for 10.3% of unexpected flows.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
RTMP	47.7	1.79	96	301
Unknown	15.0	16.3	5,835	3,681
Taobao	13.4	0.49	8	269
Spotify	11.4	1.93	113	2,039
SPDY	3.70	1.74	73	628
WeChat	3.35	8.18	47	3,055
Invalid HTTP	2.56	2.36	731	723
HTTPS	0.68	5.37	598	3,976
BadBaidu	0.05	38.6	2,594	1,832
NortonBackup	<0.01	10.9	4	52
Other (36)	2.36	12.4	-	-

Table 3: Non-HTTP Applications observed using TCP port 80 in the Waikato traces.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
RTMP	77.3	2.37	82	1,587
HTTP	5.92	10.8	386	3,918
Spotify	4.65	1.86	118	2,100
WeChat	4.35	12.5	66	3,258
Runescape	3.30	0.19	73	50
Unknown	2.04	17.0	3,385	4,754
CrashPlan	1.22	0.05	13	36
Telegram	0.23	1.84	19	230
Taobao	0.20	1.18	79	552
WhatsApp	0.20	13.7	400	3,288
Line	0.11	6.13	22	1,463
Other (33)	0.44	30.4	-	-

Table 4: Non-HTTPS Applications observed using TCP port 443 in the Waikato traces.

There was a much larger variety of application protocols observed on TCP port 80; in total, 46 unique application protocols were reported by libprotoident as using TCP port 80. Table 3 shows the major contributors of unexpected traffic on TCP port 80. RTMP (i.e. Flash) was the biggest contributor by traffic volume, producing nearly half of the non-HTTP traffic observed on TCP port 80. Traffic associated with the Chinese online marketplace Taobao (or its parent company Alibaba) comprised a further 3.4 GB of traffic, closely followed by the music-streaming application Spotify with 2.9 GB. SPDY, an application protocol developed by Google for faster delivery of web content, is responsible for nearly 1 GB of TCP port 80 traffic, much less than the 1.4 TB of QUIC traffic seen on UDP port 443.

Another protocol that we observed on TCP port 80 that is worth mentioning is the 'BadBaidu' protocol, which is associated with use of the Baidu web browser and has some peculiar characteristics. Periodically, the host running the browser would make an outbound TCP connection to one of a large number of Baidu-controlled servers. Once the handshake was complete, the client would then send a FIN ACK packet which received no response from the server. The client would typically retransmit this packet five times, before sending a packet with a single byte of payload (0x00) and an invalid sequence number[2]. The server will then reset the connection, presumably because of the invalid TCP behavior.

The purpose of these flows is not entirely clear, but the effect was apparent in our results: we observed over 150,000 flows from 1840 different unique IP addresses that matched the BadBaidu behavior in our one-week long dataset. Because each flow only generated one byte of application payload, the traffic produced by this protocol was tiny; if we had focused on byte contributions alone, this protocol would have gone unnoticed.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
Shoutcast	34.1	0.08	2	2
RTMP	27.0	54.7	7	85
Unknown	22.7	12.7	117	242
BitTorrent	7.33	0.57	9	11
QQ	3.86	31.0	17	362
Other (6)	5.01	0.87	-	-

Table 5: Non-HTTP Applications observed using TCP port 8000 in the Waikato traces.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
Xunlei	64.3	5.92	506	19
Xunlei VIP	28.2	0.81	222	23
Speedtest	5.18	0.23	7	21
Unknown	1.08	27.4	737	2,775
WeChat	0.76	28.6	43	3,068
QQ	0.35	7.70	147	1,288
SSL/TLS	0.04	0.29	15	14
RTMP	0.03	11.6	11	221
Weibo	0.02	10.9	9	1,213
Other (16)	<0.01	6.52	-	-

Table 6: Non-HTTP Applications observed using TCP port 8080 in the Waikato traces.

RTMP was an even larger contributor of unexpected traffic on TCP port 443, as shown in Table 4. 69 of the 89 GB of unexpected traffic was attributed to RTMP. The next biggest contributor was unencrypted HTTP with 5.3 GB. Spotify and WeChat (a popular Chinese messaging application) appear in Table 4, having previously been prominent on TCP port 80 as well. Other messaging applications such as Telegram, WhatsApp and Line produced large numbers of unexpected flows, suggesting that TCP port 443 is frequently used for encrypted messaging. This is probably due to two factors: firstly, TCP port 443 is unlikely to be blocked by operators and, secondly, traffic on port 443 is expected to

[2]Specifically, the sequence number matches the number consumed by the original outgoing SYN packet.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
Youku	95.2	13.0	8,734	17
Avast DNS	1.70	14.2	94	82
Wolfenstein	1.69	<0.01	2	1
360.cn	0.66	36.1	160	682
WeChat	0.29	0.02	3	5
Fortinet	0.24	3.14	22	1
Unknown	0.18	31.4	1,674	367
Other (11)	0.01	2.19	-	-

Table 7: Non-DNS Applications observed using UDP port 53 in the Waikato traces.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
WeChat	92.6	0.81	9	261
Wolfenstein	3.67	0.02	20	1
Unknown	3.57	4.51	419	751
BitTorrent	0.35	13.9	1,092	492
QQLive	0.26	31.9	22,815	136
Gnutella	0.13	47.3	13	165
Other (10)	0.36	1.57	-	-

Table 8: Applications observed using UDP port 80 in the Waikato traces.

be encrypted in some way so will not attract as much scrutiny from DPI software.

Table 5 shows that TCP port 8000 was not heavily utilized by unexpected applications. The two biggest contributors, RTMP and Shoutcast (a media broadcasting application), contributed just 28 MB of traffic combined and there were only nine Shoutcast flows on TCP port 8000 in total. Another application worth mentioning was QQ, a Chinese messaging application, which was responsible for 31% of the unexpected flows.

By contrast, TCP port 8080 featured a large amount of non-HTTP traffic, most of which was attributed to Xunlei, a file download client that is popular in China. Xunlei implements a custom protocol for P2P downloads which accounts for 32 GB of unexpected traffic on TCP port 8080 (as seen in Table 6). The Xunlei client also features the ability to accelerate a download using a VIP service. This service allows users to download parts of their file from copies stored on well-connected servers hosted by Xunlei. The accelerated download uses a separate protocol, which we have called Xunlei VIP. Speedtest, the broadband speed test developed by Ookla, is another major contributor, accounting for 2.6 GB of observed traffic on TCP port 8080.

3.2 UDP Ports

As with TCP port 53, Youku is the biggest source of unexpected traffic on UDP port 53. Aside from Youku, security-related applications appear frequently in Table 7: Avast DNS, 360.cn and Fortinet, for instance. Avast DNS refers to the 'Secure DNS' replacement for regular DNS that is included in some Avast security software. We suspect that the 360.cn traffic is a similar service for users of 360 Total Security from Qihoo.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
Cisco VPN	81.3	<0.01	2	2
Unknown	10.4	0.05	67	172
Avast DNS	5.57	28.0	94	91
OpenVPN	2.54	0.01	3	4
Skype	0.13	69.4	552	3,025
Other (8)	0.09	2.11	-	-

Table 9: Non-QUIC Applications observed using UDP port 443 in the Waikato traces.

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
QQ	60.7	28.5	506	699
Kugou	27.1	24.9	32,784	107
Unknown	8.17	12.6	478	735
QQ Live	1.65	5.89	74	564
QQPCMgr	1.13	18.0	19	329
Xunyou	<0.01	8.48	194	39
Other (11)	1.23	1.68	-	-

Table 10: Applications observed using UDP port 8000 in the Waikato traces.

For UDP port 80, 92.6% of the unexpected bytes were attributed to WeChat, although there were only 735 flows observed, as shown in Table 8. File sharing applications, including BitTorrent, QQLive and Gnutella, were responsible for almost all of the unexpected flows on UDP port 80 but the small overall traffic volume produced by these protocols suggest that UDP port 80 was being used for P2P network maintenance.

Once again, a single application dominates the unexpected bytes for UDP port 443; in this case, Table 9 shows that the Cisco SSL-based VPN protocol was responsible for 81.3% of the traffic volume. Interestingly, this traffic was produced by just two UDP flows. Skype contributed the most unexpected flows on UDP port 443 at 69.4%, but the large number of flows resulted in only 3 MB of traffic volume. Avast DNS was also prominent on UDP port 443: in this case, the traffic was conventional DNS but sent to Avast servers that are listening on UDP port 443.

UDP port 8000 was almost entirely used by Chinese applications, as shown in Table 10. The biggest by traffic volume was Tencent QQ, which had also been prominent on TCP port 8000. The other major contributor of unexpected bytes was Kugou, a music streaming application. QQ Live (video streaming), QQ PC Manager (security / anti-virus) and Xunyou (game acceleration) were large contributors of flows on UDP port 8000.

Finally, Table 11 shows that there were two applications that were responsible for most of the traffic on

Protocol	Bytes (%)	Flows (%)	Remote IPs	Local IPs
OpenVPN	83.1	0.25	56	20
WeChat	16.1	2.43	11	254
Xunlei	0.46	0.11	13	9
BitTorrent	0.32	9.16	982	287
Kankan	0.06	71.1	15	83
Unknown	0.01	12.7	151	91
Other (9)	0.01	4.31	-	-

Table 11: Applications observed using UDP port 8080 in the Waikato traces.

UDP port 8080: OpenVPN (which contributed 31.4 GB) and WeChat (which produced 6.1 GB). As with the Cisco VPN flows on UDP port 443, the number of OpenVPN flows was relatively small for the amount of traffic observed, suggesting that the VPNs were being used to transfer large files. Kankan, a P2P video streaming application from the same company as Xunlei, was the biggest contributor of flows on UDP port 8080. Based on the traffic volume observed, the Kankan traffic was most likely to be used for P2P network maintenance rather than streaming the videos themselves.

4. RELATED WORK

The increasing disassociation between port numbers and the applications that use them has been widely studied and accepted with the research community, including [6] [11] [12] to name but a few well-known publications. Most effort in this area has been concerned with finding new ways to identify applications (particularly P2P applications) that do not rely on port numbers. Deep or lightweight packet inspection have been popular approaches [1] [7] and there has also been a significant amount of research into methods based on statistical analysis and machine-learning [14] [16] [18]. Application mixes across all ports on a network have also been well studied in the past, with [3] and [15] being two notable recent studies in this space.

More specifically, there have been several previous works have identified non-HTTP applications that make use of TCP port 80. In 2010, Dainotti et al. [5] demonstrated a successful trained method for distinguishing between HTTP and non-HTTP traffic on TCP port 80. The authors found that approximately 5% of TCP port 80 packets were classified as non-HTTP. Much of this traffic was further identified as P2P traffic but the authors were unable to identify the application protocol for nearly half of the non-HTTP flows that they observed, which suggests there may have been many other unidentified applications that were also using TCP port 80 at that time.

Gebert et al. [10] published an analysis in 2012 of traffic from a German broadband wireless provider using OpenDPI (the forerunner to nDPI) and found that a large proportion of the traffic that OpenDPI was unable to identify appeared on TCP port 80. They also found significant quantities of BitTorrent and MSN Messenger traffic on TCP port 80. TCP port 443 is also mentioned as a contributor of unidentified traffic in the paper but there is no mention of the applications that might be responsible for that traffic. In 2006, Dreger et al. [8] recognized that applications could use TCP port 80 to bypass firewalls and published a table showing that there is a proportion of flows on TCP port 80 that did not match the signature for HTTP included with l7-filter.

Maier et al. [13] examined the relationship between TCP port 80 and HTTP traffic in 2009 and found that 98.1% of the bytes on TCP port 80 were HTTP, which correlates reasonably well with our own findings, and that Shoutcast was the largest non-HTTP application on TCP port 80. More recently, Richter et al. [15] reported that less than 0.3% of TCP port 80 traffic observed at a European IXP in 2013 did not involve a known HTTP endpoint, therefore they assumed that the traffic was most likely not HTTP.

An important distinction between our work and these previous studies is the greater depth in our analysis of the unexpected traffic. We have taken the time to investigate and identify unknown application protocols so that we are able to account for the majority of the traffic we are studying. Furthermore, whereas only [10] looked beyond TCP port 80, we have investigated other port numbers that are associated with popular Internet applications, such as port 53, 443 and 8080.

5. CONCLUSION

This paper presents the results of an investigation into unexpected traffic on the TCP and UDP ports that are associated with core Internet applications such as HTTP, HTTPS, DNS and NTP. The aim of this work was to identify and quantify applications that may be co-opting these ports to potentially bypass traditional port-based traffic controls, such as firewalls.

This study examined traffic captured at the border of the University of Waikato campus network in September 2015 and used the libprotoident software to categorize the traffic based on the application protocol being used. We researched payload patterns that libprotoident was initially unable to identify and developed new rules for many previously unknown application protocols. These rules have been included in the latest release of libprotoident both for the benefit of other researchers and to demonstrate that these applications can easily be recognized through lightweight packet inspection.

While we found that in many cases the expected application protocol accounted for the vast majority of the flows and bytes observed on the ports that we investigated, the absolute quantity of unexpected traffic on some ports was not negligible. While this is not an unprecedented result in itself, further investigation of the unexpected traffic using libprotoident revealed some interesting trends, particularly the large number

of Chinese applications using ports that are commonly associated with other applications. We also discovered some strange behavior associated with the Baidu web browser that was responsible for a huge number of unexpected flows on TCP port 80, each producing just a single byte of application payload. The purpose of these flows remains unclear and this is one obvious avenue for further investigation.

One caveat is that the predominance of Chinese applications in our results may be a specific quirk of our dataset; 8.4% of students at the University of Waikato are visitors from either China or Taiwan and the usage of Chinese applications might not be as notable at other campuses with different demographics. For this reason, we present this paper more as an interesting data point that can be compared against by future studies rather than a definitive result that can be generalized across all networks.

However, this does not invalidate our finding that these applications exist and are making use of the port numbers noted in this paper; it simply means that the relative impact of the Chinese applications may not be as large on other networks, depending on the demographics of the user population. As we have published the protocol identification rules that we used for this study, we anticipate that researchers with access to other datasets (including residential and corporate networks) will repeat our analysis to determine which of our findings can be generalized and which are specific to our dataset.

6. REFERENCES

[1] S. Alcock and R. Nelson. Libprotoident: Traffic Classification using Lightweight Packet Inspection. *WAND Network Research Group, Tech. Rep*, 2012.

[2] S. Alcock and R. Nelson. Measuring the Accuracy of Open-source Payload-based Traffic Classifiers using Popular Internet Applications. In *Local Computer Networks Workshops (LCN Workshops), 2013 IEEE 38th Conference on*, pages 956–963. IEEE, 2013.

[3] V. Carela-Español, P. Barlet-Ros, A. Bifet, and K. Fukuda. A Streaming Flow-Based Technique for Traffic Classification Applied to 12 + 1 Years of Internet Traffic. *Telecommunication Systems*, pages 1–14, 2015.

[4] V. Carela-Español, T. Bujlow, and P. Barlet-Ros. Is our Ground-Truth for Traffic Classification Reliable? In *Passive and Active Measurement*, pages 98–108. Springer, 2014.

[5] A. Dainotti, F. Gargiulo, L. I. Kuncheva, A. Pescapè, and C. Sansone. Identification of Traffic Flows Hiding Behind TCP Port 80. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[6] A. Dainotti, A. Pescape, and K. C. Claffy. Issues and Future Directions in Traffic Classification. *Network, IEEE*, 26(1):35–40, 2012.

[7] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano. nDPI: Open-source High-speed Deep Packet Inspection. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International*, pages 617–622. IEEE, 2014.

[8] H. Dreger, A. Feldmann, M. Mai, V. Paxson, and R. Sommer. Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection. In *USENIX Security*, pages 257–272, 2006.

[9] J. Fan, J. Xu, and M. H. Ammar. Crypto-PAn: Cryptography-based Prefix-preserving Anonymization. *Computer Networks*, 46(2), 2004.

[10] S. Gebert, R. Pries, D. Schlosser, and K. Heck. Internet Access Traffic Measurement and Analysis. In *Proceedings of the 4th International Conference on Traffic Monitoring and Analysis*, pages 29–42. Springer-Verlag, 2012.

[11] T. Karagiannis, A. Broido, N. Brownlee, K. Claffy, and M. Faloutsos. Is P2P Dying or Just Hiding? In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, volume 3, pages 1532–1538. IEEE, 2004.

[12] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 229–240. ACM, 2005.

[13] G. Maier, A. Feldmann, V. Paxson, and M. Allman. On Dominant Characteristics of Residential Broadband Internet Traffic. In *Proc. of the 9th ACM Internet Measurement Conference*, pages 90–102. ACM, 2009.

[14] T. T. Nguyen, G. Armitage, P. Branch, and S. Zander. Timely and Continuous Machine-Learning-based Classification for Interactive IP Traffic. *IEEE/ACM Transactions on Networking (TON)*, 20(6):1880–1894, 2012.

[15] P. Richter, N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger. Distilling the Internet's Application Mix from Packet-Sampled Traffic. In *Passive and Active Measurement*, pages 179–192. Springer, 2015.

[16] A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura, and A. Nucci. Towards Self Adaptive Network Traffic Classification. *Computer Communications*, 56:35–46, 2015.

[17] WAND Network Research Group. *WDCap*, 2016. http://research.wand.net.nz/software/wdcap.php.

[18] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu. Robust Network Traffic Classification. *IEEE/ACM Transactions on Networking*, 23(4):1257–1270, 2015.