How device misconfiguration drives TCP traffic to parts of 1.0.0.0/8 – an initial investigation

Mattia Rossi, Grenville Armitage, Geoff Huston
Centre for Advanced Internet Architectures, Technical Report 110720A
Swinburne University of Technology
Melbourne, Australia
mrossi@swin.edu.au, garmitage@swin.edu.au, gih@apnic.net

***Abstract*—The Internet community is near the 'bottom of the barrel' for unallocated IPv4 address prefixes. Network 1.0.0.0/8 was allocated in January 2010 for use on the public Internet, despite being unofficially utilised in various ways for many years. Recent work has revealed this prefix to be quite 'dirty', with significant levels of public UDP and TCP traffic already inbound to certain parts of 1.0.0.0/8. By running a simplified honeypot on 1.1.1.0/24 and 1.2.3.0/24 for two days in March 2010 we have elicited new insights into the nature of the TCP traffic polluting these prefixes. Our honeypot replied to inbound TCP SYN packets with a SYN-ACK, thereby eliciting a variety of subsequent response packets from sources actively trying to connect into 1.1.1.0/24 or 1.2.3.0/24 space. By analyzing captured packet payloads, sequences, retransmission patterns and burst rates within such TCP flows, we find that most TCP traffic into these prefixes is caused by some form of misconfiguration rather than malice, and we discuss the possible causes for these misconfigurations.**

I. Introduction

IPv4 network 1.0.0.0/8 has been in the unallocated pool of addresses for many years, and in January 2010 was allocated to APNIC (Asia Pacific Network Information Centre) for use in allocation of addresses for the public Internet [1].

Parts of 1.0.0.0/8 have been unofficially used as internal or testbed address space by many organizations over the years. Recent studies have shown that anyone advertising routes to certain /24 subsets of 1.0.0.0/8, such as 1.1.1.0/24, 1.2.3.0/24 and 1.0.0.0/24, would be flooded with tens to hundreds of Mbit/sec of uninvited UDP traffic, rendering those /24 prefixes unsuitable for assignment to end users [2], [3], [4]. TCP traffic was also observed (mainly TCP SYN packets and a small percentage of misrouted TCP DATA packets), but the passive capture techniques allowed no further insight into the nature of the TCP sources.

Our work extends previous studies of 1.1.1.0/24 and 1.2.3.0/24 traffic by acting as a simplified honeypot [5], responding to TCP SYN packets with a specially crafted SYN-ACK to elicit further packets from each source. We study the subsequent reactions of each TCP source to differentiate between sources controlled by malware (for example, someone specifically doing portscans of 1.0.0.0/8 space, etc) and sources who reach 1.0.0.0/8 space through misconfiguration close to the source itself.

This paper studies TCP traffic to 1.1.1.0/24 and 1.2.3.0/24 collected during a brief period in mid-March 2010. We acted as a simplified honeypot for 10 hours and observed 54.3M connection attempts (unique initial TCP SYN packets) into 1.1.1.0/24, and 13.5M connection attempts into 1.2.3.0/24. Not all TCP sources responded to our SYN-ACKs – we observed a 71% response rate from sources connecting into 1.1.1.0/24, and 78% for sources connecting into 1.2.3.0/24. Responsive sources behaved in a variety of ways. Many moved to the next stage in their TCP state machine and attempted to send us data, others simply closed their connections after differing periods of time.

By evaluating patterns of behavior by responsive sources we come to the following conclusion: sources that attempt to establish a genuine TCP connection into 1.1.1.0/24 or 1.2.3.0/24 generally do so due to misconfiguration somewhere close to the source (rather than because the source actively intends to probe destinations in 1.1.1.0/24 or 1.2.3.0/24 space). This includes the use of equipment configured to use 1.0.0.0/8 in a private context where the local firewalls have failed to contain the traffic, or the pre-configured device has been relocated from a private environment to a public context and the configuration cause the device to "leak" traffic to the public address. Malicious probing or scanning contributes far less than to the pollution of 1.1.1.0/24 or 1.2.3.0/24 space than simple human misconfiguration.

The paper is structured as follows: Section II summarizes previous work on traffic seen heading towards 1.0.0.0/8. Section III details our own collection, and our TCP traffic analysis occurs in Section IV. We conclude in Section V.

II. Related work

Shortly after 1.0.0.0/8 was allocated to APNIC, concerns arose about the level of unsolicited traffic that would be received by anyone advertising subsets of that prefix [6]. Initial trial announcements of parts of 1.0.0.0/8 by RIPE-NCC in collaboration with APNIC confirmed the concerns, with the inbound traffic saturating the 10 Mbit/sec link of the router advertising 1.1.10/24 and 1.2.3.0/24 [2]. About 90% of the traffic was addressed to 1.1.1.1, while 4% of the traffic was addressed to 1.2.3.4. Furthermore, 81% of packets were UDP, 14% were TCP and 4% ICMP. Of the TCP packets, 49% appeared to be HTTP connection attempts (SYN packets to port 80), while 1% appeared to be "established" HTTP connections (non-SYN packets to port 80).

APNIC collaborated with Merit Networks (AS 237) and YouTube (AS 36351) on a second trial, where the whole 1.0.0.0/8 prefix was announced at AS237 from 22 February 2010 until 1 March 2010, then at AS 36351 for 6 hours on March 21 [3]. The trial was dominated by traffic inbound to 1.1.1.0/24 (around 90 - 100 Mbit/sec), with 1.1.1.1 the main target address. Only 233 out of 65536 /24 prefixes (0.4%) of the 1.0.0.0/8 block, attract more than 3 packets per seconds (and more than 3Kbps). The remaining 99.6% of the /24s within 1.0.0.0/8 could be considered "clean" as they experienced very little traffic. The three most polluted /24 prefixes were 1.1.1.0/24, 1.2.3.0/24 and 1.0.0.0/24. Consistent with the RIPE-NCC trial [2], the majority of inbound packets were UDP, the majority of TCP packets were TCP SYN requests, and the few remaining TCP packets were identified as misrouted packets from established TCP connections.

Further research [4] announced (and captured traffic to) a number of unallocated /8 blocks, and showed that far more unsolicited traffic heads toward 1.0.0.0/8 than the (at the time unallocated) 35.0.0.0/8, 50.0.0.0/8 and 107.0.0.0/8 ranges. UDP traffic levels heading to 1.0.0.0/8 were similar to that seen in earlier work. However, [4] went further and correctly identified a high volume of UDP traffic as RTP streams carrying one-way audio of a female voice saying "The number you have dialed is not in service, please check the number and try again". We have observed this kind of behaviour when, for example, an insecure trixbox VoIP server [7] is subjected to a SIP INVITE scan. A misconfigured INVITE message could easily direct the resulting RTP stream at a target in 1.0.0.0/8.

To date, no-one has studied the behaviour of sources attempting to create TCP connections into 1.0.0.0/8 space.

III. Methodology

Due to being identified as significantly polluted members of 1.0.0.0/8 [3] we chose to focus on collecting traffic destined for 1.1.1.0/24 and 1.2.3.0/24.

A. Data Collection

Data collection involved a locked-down FreeBSD host connected via 1Gbit/sec Ethernet to the Australian Academic and Research Network (AARNet) from March 17th to March 19th 2010. At 10:00hr March 17th (UTC) we announced AARNet (AS 7575) as a route to both 1.1.1.0/24 and 1.2.3.0/24, and began collecting all inbound IP packets with full payloads. We replied to all inbound TCP SYN packets with a SYN-ACK. Each SYN-ACK contained an artificial sequence number so we could identify which subsequent inbound TCP packets were triggered by particular SYN-ACKs. (The sequence number was constructed from a source's IP address, bit-wise reversed and incremented by one.) At 20:10 on March 17th (UTC) we stopped responding to TCP SYN packets, but continued passively collecting all inbound IP traffic until 08:00 on March 19th.

In short, we actively triggering further responses from remote TCP sources during the first 10 hours. The subsequent 36 hours of passive traffic collection is used for comparison with prior work.

B. Initial observations and post-processing

Our initial analysis unexpectedly revealed a stream of TCP SYN packets (20% of all TCP SYNs received over 46 hours) purporting to come from 1.0.0.0/8 space, always targeting port 1 of their destination, with TTL values spread uniformly between 106 and 239, and source addresses scattered across 99.8% of the 1.0.0.0/8 space. As the originators could not have expected replies to their forged (and officially unroutable) source addresses, we eliminated these packets from further consideration.

Figure 1 shows the total data rate (all packets, regardless of protocol) to both prefixes after eliminating the traffic purporting to be from 1.0.0.0/8. There is a clear drop at the 10hr mark when we stopped responding with SYN-ACK packets, and thus stopped triggering further

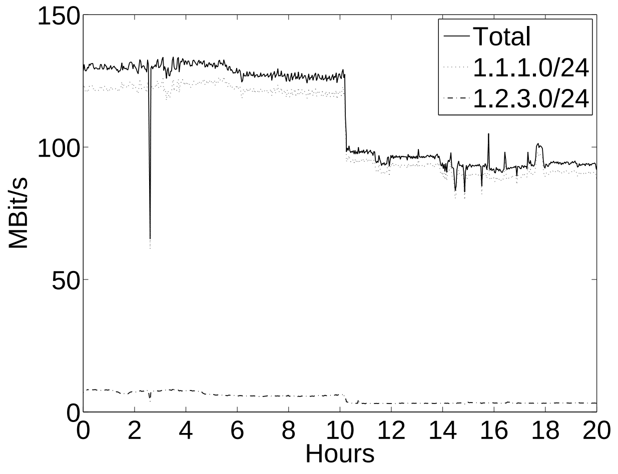
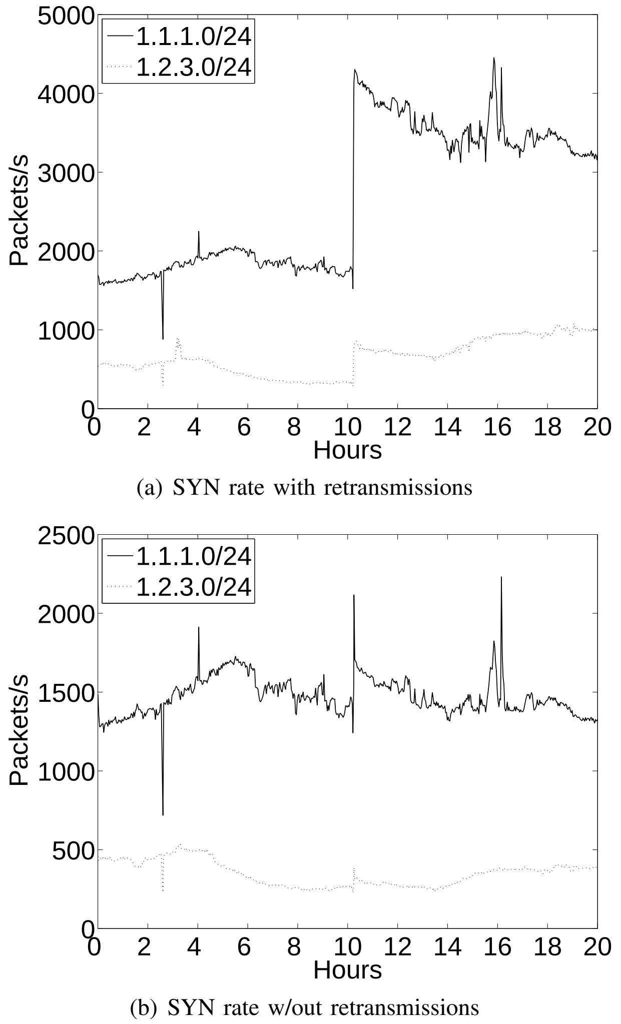


Fig. 1. Inbound traffic rate during first 20 hours – total traffic, and traffic to 1.1.1.0/24 and 1.2.3.0/24 separately. We stop sending SYN-ACKs at 10hr mark.

TCP packets from responsive sources. After the 10hr mark our results mimic that of previous research that passively collected traffic to 1.1.1.0/24 and 1.2.3.0/24 (e.g. [3]). Consistent with prior work, traffic to 1.1.1.0/24 significantly outweighing traffic to 1.2.3.0/24 at all times. This gives us some confidence that inbound TCP SYN traffic during the first 10 hours is also likely to have been 'typical'.

TCP traffic collected passively (such as the experiments in Section II) contain both initial SYN packets (from sources just beginning connection establishment) and retransmitted SYN packets (from sources whose initial SYN packet has not elicited a response from their destination). Figure 2(a) drills down to show the TCP SYN traffic that we received before and after the 10hr mark.

Prior to the 10hr mark, our SYN-ACK replies substantially eliminate retransmission of SYNs by standards-compliant sources. After the 10hr mark there is a spike in aggregate SYN traffic – the flow of new initial SYN packets being supplemented by retransmissions of previous SYNs that are now being ignored. Figure 2(b) shows that, after eliminating retransmitted SYNs[1], replying with a SYN-ACK (or not) has no significant impact on the rate of initial SYNs.



(a) SYN rate with retransmissions

(b) SYN rate w/out retransmissions

Fig. 2. TCP SYN packets/sec received over the first 20 hours by 1.1.1.0/24 and 1.2.3.0/24, averaged every 2 minutes

Prefix 1.1.1.0/24			Prefix 1.2.3.0/24		
Address	Init. SYNs	% of total	Address	Init. SYNs	% of total
1.1.1.1	183358388	82.9%	1.2.3.4	49300508	82.0%
1.1.1.2	11618081	5.3%	1.2.3.250	3403628	5.7%
1.1.1.4	9194874	4.2%	1.2.3.13	1061002	1.8%
1.1.1.3	8981316	4.1%	1.2.3.11	913787	1.5%
1.1.1.5	1222798	0.6%	1.2.3.12	893395	1.5%

TABLE I
TOP 5 DESTINATION ADDRESSES FOR INITIAL SYN PACKETS OVER 46 HOURS (PERCENTAGE RELATIVE TO TOTAL COUNT OF INITIAL SYNS)

IV. DETAILED ANALYSIS OF TCP CONNECTION ATTEMPTS

A. *These are not address or port scans*

The first thing we notice from inbound TCP SYN traffic is the limited evidence of 'scanning'. Instead, the inbound TCP connection attempts (initial SYNs) target a small range of IP addresses and ports. Table I shows the top 5 most targeted addresses and Table II shows the top 10 most targeted ports across the entire 46 hours of packet collection for 1.1.1.0/24 and 1.2.3.0/24. Both tables count only initial SYNs.

Roughly 83% of new connection attempts into 1.1.1.0/24 target address 1.1.1.1, and 82% of new con-

[1] Which we defined as one or more SYNs in a 2 minute window for the same source and destination pair, and same initial sequence number

1.1.1.0/24			1.2.3.0/24		
Port	Init. SYNs	% of total	Port	Init. SYNs	% of total
80	53192548	24.0%	80	24566078	40.9%
6112	39248168	17.7%	82	12482184	20.8%
5022	23022143	10.4%	8888	5618099	9.3%
443	15243794	6.9%	25	2575354	4.3%
6667	10698972	4.8%	4201	2375348	3.9%
25	6330188	2.9%	7777	1309942	2.2%
3175	6326319	2.9%	443	801374	1.3%
502	4119181	1.9%	6969	771991	1.3%
110	3906095	1.8%	6112	486364	0.8%
6969	3221953	1.5%	8080	444187	0.7%

TABLE II
TOP 10 DESTINATION PORTS FOR INITIAL SYN PACKETS OVER 46 HOURS (PERCENTAGE RELATIVE TO TOTAL COUNT OF INITIAL SYNS)

nections into 1.2.3.0/24 target address 1.2.3.4 – addresses that are symbolic and easy to remember in dotted-quad form. The top five addresses account for 97% and 92% of all new connections into 1.1.1.0/24 and 1.2.3.0/24 respectively. New connections are attempted to 28 433 and 10 323 different ports across 1.1.1.0/24 and 1.2.3.0/24 respectively. However, only 10 destination ports account for roughly 75% and 85% of initial TCP SYN packets sent to 1.1.1.0/24 and 1.2.3.0/24 respectively. Taken together with the very narrow selection of observed destination IP addresses, we believe no one is doing comprehensive address and port scans within either of those prefixes.

B. Responses to SYN-ACKs

Our SYN-ACKs might elicit no response for a variety of reasons. A source's response packet might simply be lost, our SYN-ACK itself might be lost, or the source might simply be probing with SYNs and having no intention of establishing or closing a connection in a standards-compliant manner. Losses might be due to congestion in either direction, or filters somewhere along the path objecting to traffic to or from anywhere in 1.0.0.0/8[2].

Nevertheless, the majority of SYN-ACKs elicited a further response[3]. Figure 3 shows the arrival rate of initial SYNs towards 1.1.1.0/24 and 1.2.3.0/24 during the first 14 hours, and arrival rate of responses to our SYN-ACKs during the first 10 hours. Response rates seem to be a fairly constant fraction of initial SYNs. Our SYN-ACKs elicited 38.8M responses from 54.3M connection attempts to 1.1.1.0/24 and 10.5M responses from 13.5M connection attempts to 1.2.3.0/24.

[2]For example, an asymmetric firewall configuration mistakenly allowing 'private' 1.0.0.0/8 traffic to leak out but correctly blocking inbound traffic towards internal privately addressed devices

[3]Unique responses, ignoring retransmitted SYN packets and ACK, FIN, RST or DATA packets which don't seem to be part of any TCP connection attempt

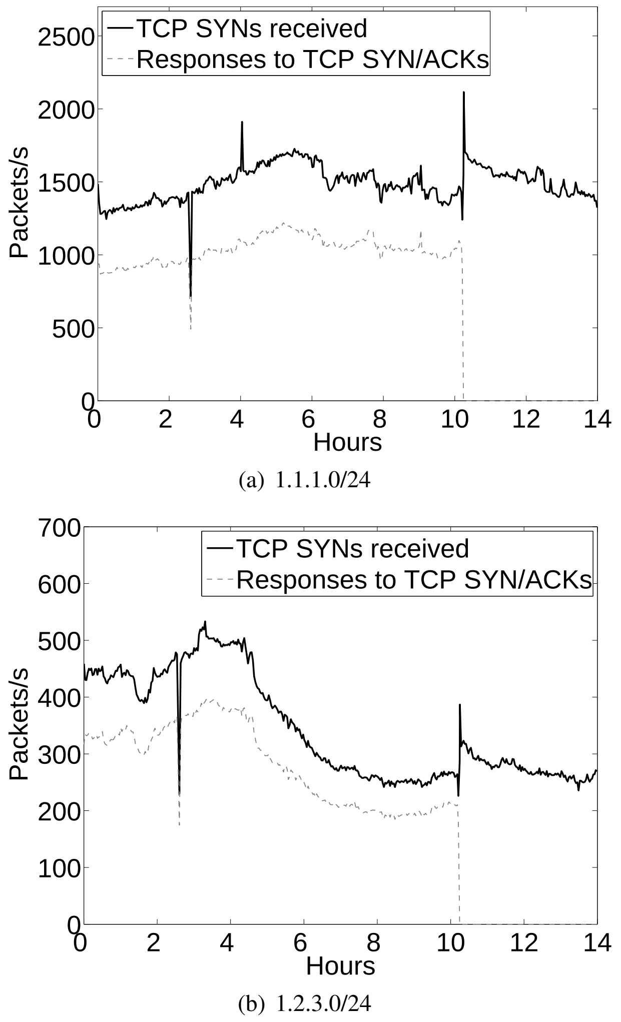


Fig. 3. TCP SYN packets and packets elicited by a TCP SYN-ACK over 10 hours

Around 55% of connection attempts to addresses 1.1.1.1 and 1.2.3.4 (the top most targeted addresses in each prefix) responded to our subsequent SYN-ACK. Table III shows the variation in response rates for connection attempts to the top 10 ports in each prefix.

1.1.1.0/24			1.2.3.0/24		
Port	Init. SYNs	Response rate	Port	Init. SYNs	Response rate
80	14956761	66.06%	80	6923442	62.42%
5022	10706960	68.91%	82	3675359	84.86%
6112	10106443	28.11%	25	876805	66.15%
443	4043301	83.93%	4201	833548	55.05%
6667	2403285	21.91%	8888	488026	55.29%
25	2344910	53.44%	6969	465967	6.85%
110	1848456	29.87%	443	258555	73.07%
6969	1280985	74.84%	8000	243444	19.49%
6881	1277988	95.97%	8080	228592	31.96%
3175	1080826	12.92%	6112	216078	70.19%

TABLE III
NUMBER OF CONNECTION ATTEMPTS TO TOP 10 PORTS OVER FIRST 10 HOURS, AND PERCENTAGE OF RESPONSES ELICITED BY OUR SYN-ACKS.

C. Retransmission of responses

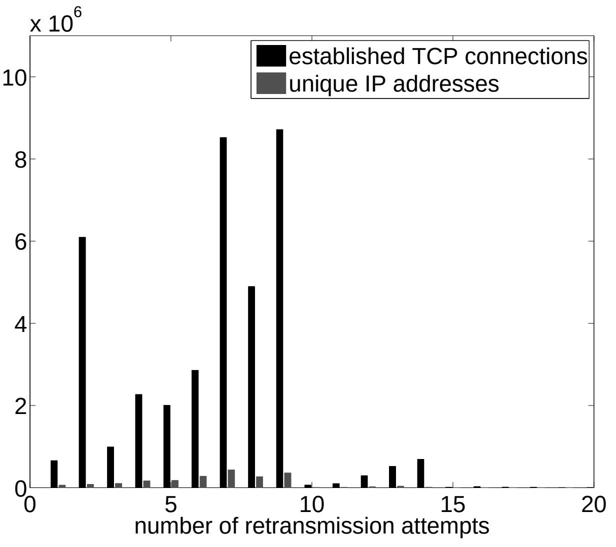
Most of the responses to our SYN-ACKs are retransmissions caused by the lack of sending ACKs from the server back to the clients. Figure 4 shows the number of TCP connections and the number of source IP addresses per number of retransmission attempts. While most of the connections to either 1.1.1.0/24 or 1.2.3.0/24 time out between 1 and 9 packet retransmissions, there is a noticeable peak of sources that retransmits packets between 11 to 14 times. A further, less evident peak can be found between 15 and 20 retransmissions. We also detected TCP connections with abnormal retransmissions scattered from 21 up to 485 times in 1.1.1.0/24 and from 21 to 583 times in 1.2.3.0/24, all caused by a small set of 20 to 30 different source addresses. In 1.1.1.0/24 we additionally found a single host retransmitting packets up to 27727 times.

The finding suggest, that there are three distinct groups of operating systems or TCP stacks with fixed timeout values at 9, 14 and 20, and a handful of sources doing something odd. Apart from the single source retransmitting 27727 times, we also found sources reusing the same sequence numbers when resending a SYN after a retransmission timeout. It is also possible to determine from the graph, that given the lower number of unique IP addresses compared to TCP connections, single sources attempt multiple TCP connections to the site.

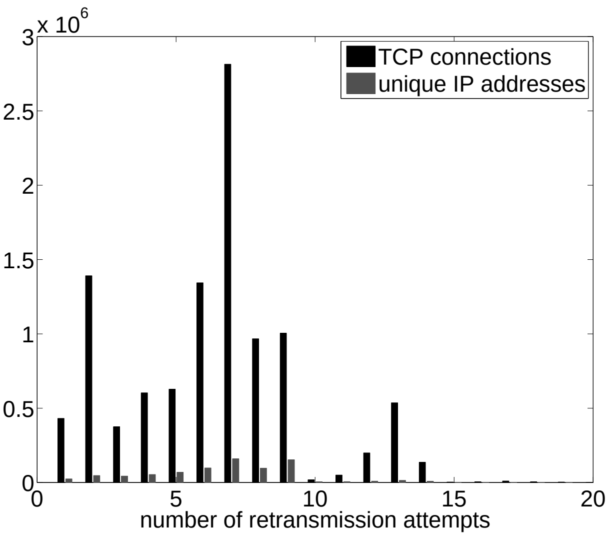
A comparison of TTL values of the SYN packet to the TTL values of the response packets of each connection resulted in 1.2M TTL value changes for prefix 1.1.1.0/24, and 346K for prefix 1.2.3.0/24 between a SYN and the response packets. Furthermore we found that TTL values changed multiple times within a retransmission sequence in 10545 cases for prefix 1.1.1.0/24, and in 1441 cases for prefix 1.2.3.0/24. Sometimes the TTL difference between SYN and response packets is quite high as shown in Figure 5. Of the 1330855 sources responding to our SYN/ACKs we found 2474 sources whose TTL changes by more than 30 between their SYN and subsequent response packets[4]. Of those sources, 98% re-connect more than once and exhibit the same change in TTL during each re-connection attempt.

We also approximated the hop count by subtracting the measured TTL value from the next highest multiple of 32. Multiples of 32 are commonly used as initial values for the TTL field in IP stacks (with the exception of 256; we considered the start value of 255 instead). Figure 6 shows the spread of the calculated hop counts between SYNs and ACKs. Comparing the TTL and hop-count graphs, we see that many changes in TTL between SYN and subsequent response packets do not always show up as a change in path length. This might occur if a middlebox close to the source rewrites the initial TTL value of either the SYN or the ACK. As we use multiples of 32 as initial TTL values, we can only detect a maximum of 32 hops. This restriction is characterized by the outliers seen at the top of the graphs in Figure 6. We suspect that some paths are even slightly longer than 32 hops, which would be displayed with a hop

[4]Small TTL differences might be caused by path changes occurring between the source's transmission of SYN and ACK, but TTL differences over 30 cannot be explained by path changes.

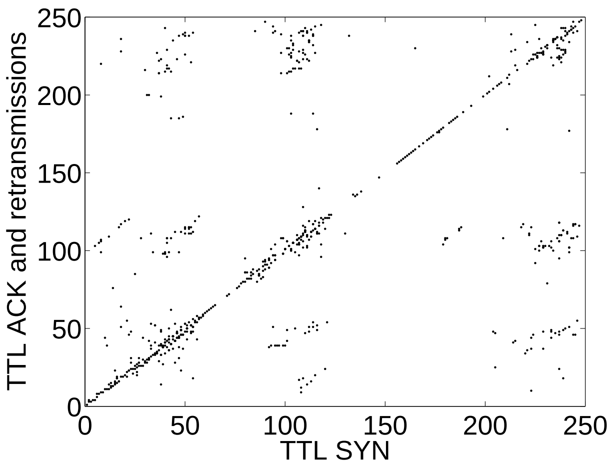


(a) 1.1.1.0/24

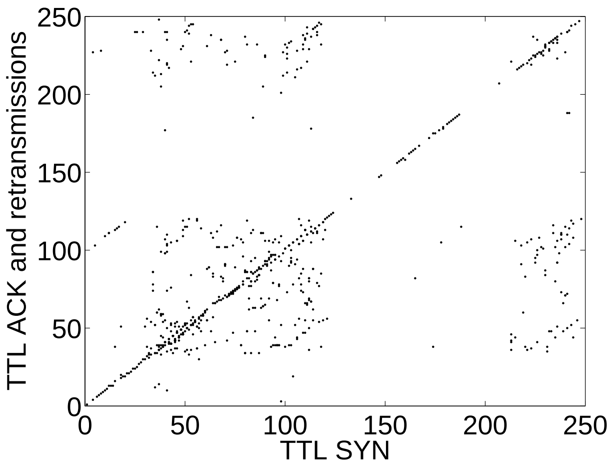


(b) 1.2.3.0/24

Fig. 4. Number of TCP connections and unique source IP addresses per number of packet retransmissions. The graph has been limited to 20 packet retransmissions for better readability

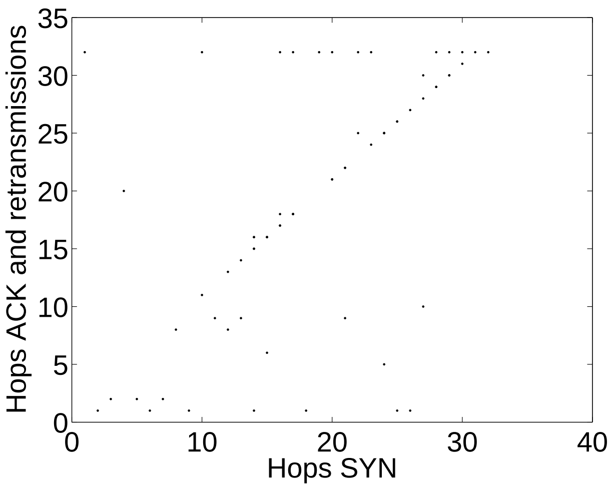


(a) 1.1.1.0/24

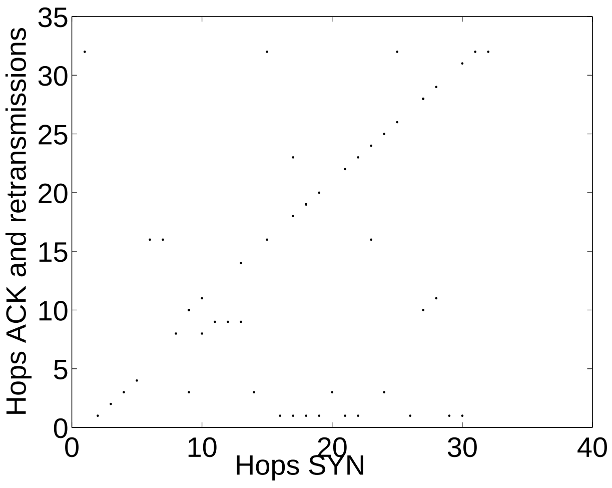


(b) 1.2.3.0/24

Fig. 5. Spread of TTL values of SYN packets and responses



(a) 1.1.1.0/24



(b) 1.2.3.0/24

Fig. 6. Spread of TTL values of SYN packets and responses

count like 1 or 2 in the graph. Paths of 32 hops or longer might seem plausible (having been seen even in 1998 [8]). However, we believe it is more likely that some hosts targeting our 1.1.1.0/24 and 1.2.3.0/24 space are using non-standard TCP stacks that exhibit distinctly unusual initial TTL values. Most of the hosts which yield paths of 32 hops or a huge difference between the pathlength of a SYN and the ACKs are also responsible for the abnormally high number of TCP retransmissions within a single connection, coupled with an abnormally high retransmission time out. Hosts with such abnormal settings have been found in small numbers throughout the dataset at almost all path lengths, but with higher concentration at very long paths (>30 hops) and very short paths (<5 hops), which increases our suspicion of the presence of paths longer than 32 hops. We doubt that such stacks use a standard initial TTL value in some cases, and we believe the initial TTL is set to different values for SYN and ACK packets.

D. *Observed response sequences*

Where sources responded to our SYN-ACKs, we observed response sequence falling into one of four categories. The following list indicates the percentage of each sequence type observed for 1.1.1.0/24 and 1.2.3.0/24 respectively:

1) [68.49%, 46.28%] TCP ACKs followed by TCP DATA packets and in cases by a RST, or FIN, or a FIN followed by an RST, including retransmission of ACKs and DATA. DATA can be piggybacked on the initial ACK, and always carries the ACK flag, while eventual RSTs and FINs are mostly piggybacked on the last retransmitted ACK (which can also carry retransmitted DATA).
2) [26.99%, 47.53%] TCP ACKs without DATA immediately followed by a FIN or RST, or a FIN followed by an RST (sometimes compressed to a single RST/ACK or FIN/ACK) including retransmissions of the ACKs. In case of ACK retransmission, the RST and FIN are pig-

gybacked on the last retransmitted ACK. In case of an RST following a FIN it's mostly without ACK. In some cases the RST/ACK or FIN/ACK are again followed by an ACK and the whole series is retransmitted.

3) $[0.015\%, 0.005\%]$ Immediate RST.
4) $[4.5\%, 6.18\%]$ TCP ACKs without DATA piggybacked and not followed by a RST or FIN. The ACKs are sometimes retransmitted, while other times they are sent multiple times with different sequence numbers.

Where an ACK is involved in a packet sequence (with or without carrying DATA, RST or FIN), the PSH flag could be set as well. In very few cases retransmissions of ACKs have the URG flag set as well.

Sequence 1 reflects sources with data to send – possibly genuine TCP connections timing out on the clients side. Sequences 2 and 3 lack data transfer, and suggest some form of port scanning or probing[5]. Sequence 4 opens a connection to the destination but sends no data. This could be malware or other software that opens a connection then waits for the other end to push content back.

We also found 4.6M connection attempts sending an ACK, followed by an RST/ACK after approximately 30 seconds. While the packet order corresponds to Sequence 2, we suspect these are also sources that open a connection then wait for the other end to push content back.

E. Examples of connections that transfer no data

Three examples cover 61% of all connection attempt/response sequences coming into 1.1.1.0/24 that fall under Sequence 2, 3 or 4 (connections carrying no data).

The first example (23% of sequences without data, 7% of the total number of responses) involves connection attempts to port 6112 where the source responds with an ACK followed by a RST/ACK in less than 2ms. It originates from 25 unique sources over the 10 hour period, with individual sources launching sustained attempts over periods from 2 to 5 hours (and the most active source averaging 64 connections/sec for 4 hours). Port 6112 seems to be used for a variety of games by Blizzard Entertainment [10]. The sustained repetition of connection setup and rapid teardown leads us to leading us to suspect these sources may be attempting to disrupt network access of a distant game client or server. We also see the same sequences from 20 sources targeting destination 1.2.3.4, covering 1.4% of the total connection attempts, and 2.7% of sequences without data to 1.2.3.0/24. Interestingly, 99% of these connection attempts arrive from a single source over roughly 6 hours.

Two more examples (37% of sequences without data, 12% of the total number of responses) involve 23611 sources sending sequences consisting of ACKs followed by a RST/ACK or a FIN/ACK 30 or 120 seconds later (with FIN/ACKs retransmitted up to 6 times). Sequences ending in RST/ACK mostly target port 80 and 443, while the sequences ending in a FIN/ACK mostly target ports 25 and 110. There are a large number of sources tending to send the same sequence continuously at fixed time intervals. We suspect the sources host a form of malware that uses well-known ports to 'call home' for pushed content. The target is usually 1.1.1.1, probably hard coded into the malware, left over from initial testing, or due to the sources being behind some form of misconfigured proxy.

The amount of connections covered by the three examples present a lower bound, as possible packet loss makes it difficult catalogue all connection into the correct sequence.

F. Non-HTTP connections that try to transfer data

Responses with DATA packets allow us to gain further insight into the possible intentions behind each connection attempt. Here we look at a number of non-HTTP examples (HTTP traffic will be analyzed in the next section).

Around 300K different source addresses were seen establishing connections to destination 1.1.1.1 port 5022, constituting 18% of all the connection attempts (and 26% of connections with data). We saw a 69% response rate to our SYN-ACKs, and inspection of the DATA payloads revealed clear text such as "MicrosKdsDisplayAppV1.0" and "Order currently cannot be completed" in every response. This strongly suggests that much of the port 5022 traffic was originating from widely deployed, yet misconfigured, instances of a "MICROS Kitchen Display System" product used in the hospitality industry [11]. We observed a continuous load of around 70 connections/sec over the entire 10 hours.

Traffic to port 6969 contains mostly HTTP style GET requests as used by Bittorrent to communicate to the tracker, with around 90% of the requests containing the same domain in the requests "host" part, but addressed to 1.1.1.1, showing an obviously misconfigured DNS entry [6]. Packets to port 6881 contain some Bittorrent

[5]For example, *Nmap* [9] port scans close connections by sending a FIN/ACK or RST packet after receiving a SYN-ACK. Nmap is not the only tool in use, as we see many sequences of ACKs followed shortly after by RST/ACKs

[6]This domain still resolved to 1.1.1.1 in December 2010 and is registered at "no-ip.info" [12], a common dynamic DNS service. A web search showed a variety of torrent files listed a Bittorrent tracker at the domain on port 6969

handshakes and what looks like packets containing large chunks of data or encrypted data (similar to packets sent to port 443). The data sent to port 6881 does not seem related to the misconfigured tracker.

At port 6667, used by the IRC protocol, we find mostly seemingly real IRC connection attempts and some non IRC traffic. The seemingly real IRC traffic were mostly initiated with a NICK and USER of the form "TsGhUSA-XP". According to various sources in the Internet, this user is attributable to the "TsGh BotNet" and various malware. Similar IRC malware connections have also been found in connections to ports 6969, 7029 and 51987.

The "t1-e1-over-ip" protocol is assigned to port 3175. It seems to be intended for encapsulating and sending audio and video from circuit switching system over IP. All connections to the port carry data, which we could not decode and could in fact be part of the "t1-e1-over-ip" protocol. About 10% of the sources connecting to port 3175 also connect to port 81 before. The content of the data to port 81 could not be decoded as well.

Port 4201 is assigned to the "vrml-multi-use" protocol, but seems to be used by the "War" Trojan as well – which we believe is the cause of the connections. It carries packets of around between 300 and 500 bytes in length, containing information about the OS, the browser and Mail client of the source, as well as it's local IPv4 address. It originates from 126482 different source addresses, many of them contiguous and attempts 3 to 4 connections per second continuously over 10 hours. We believe that in that case, the two destination addresses 1.2.3.4 (77% of connections) and 1.2.3.5 (23% of connections) are coded into the worm.

Around 1% of connection attempts to port 25 (SMTP) and 110 (POP3) responded with DATA packets. The data packets typically contained USER, PASS and QUIT requests for port 110 and QUIT, HELO, RSET, EHLO, DATA and MAIL FROM requests for port 25. A single source was responsible for a high percentage of port 25 traffic, apparently deliberately (but misguidedly) redirecting email to 1.1.1.1. The PASS requests in port 110 only contained 2 different passwords: "leak" and "h4xg4ng". The "leak" attempts were carried out by 45 source addresses, the "h4xg4ng" only by two. For "h4xg4ng", the first source was probing for 3 hours, then stopped, then the second source was probing for 1.5 hours.

1.1.1.0/24			1.2.3.0/24		
"host" field contents	Dest. addr.	% tot.	"host" field contents	Dest. addr.	% tot.
1.1.1.1	1.1.1.1	40.1%	1.2.3.4	1.2.3.4	8.6%
1.1.1.5	1.1.1.5	2.3%	1.2.3.13	1.2.3.13	7.6%
1.1.1.4	1.1.1.4	2.3%	chewinggym.com	1.2.3.4	7.5%
1.1.1.2	1.1.1.2	2.3%	www.sinfors .com.cn	1.2.3.4	6.7%
1.1.1.3	1.1.1.3	2.2%	1.2.3.10	1.2.3.10	3.70%
swupmf.adobe .com	1.1.1.1, 1.1.1.0, 1.1.1.2	1.5%	1.2.3.12	1.2.3.12	3.6%
urs.microsoft .com:443	1.1.1.1, 1.1.1.200	1.5%	1.2.3.11	1.2.3.11	3.6%
g.ceipmsn.com	1.1.1.1	1.4%	provisioning.pure -ip.com:80	1.2.3.4	3.6%
weather.service .msn.com	1.1.1.1	0.6%	1.2.3.9	1.2.3.9	3.3%
lastfm.mxmm.de	1.1.1.1	0.6%	1.2.3.6	1.2.3.6	1.8%

TABLE IV
THE 10 MOST COMMON "HOST" FIELD VALUES FOUND IN HTTP REQUESTS TO PORTS 80, 81, 82, 8000, 8080 AND 8888 AND THE IPv4 ADDRESS THEY ARE MAPPED TO (PERCENTAGE RELATIVE TO THE TOTAL AMOUNT OF CONNECTIONS CARRYING HTTP DATA IN EACH PREFIX).

G. A better look at HTTP traffic

Data is carried by 79% and 76% of the "established" connections to port 80 in 1.1.1.0/24 and 1.2.3.0/24 (20% and 31% respectively of all "established" connections). Connections to port 80 without data exhibited sequences of type 2 or 4 (Section IV-D). Although HTTP traffic was also directed to ports 81, 82, 8000, 8080 and 8888, almost 98% and 91% of the data carrying HTTP connections were initiated to port 80 within 1.1.1.0/24 and 1.2.3.0/24 respectively.

HTTP GET and POST requests were the most common HTTP data payloads received on all the observed ports and both prefixes. Well-formed HTTP requests contain at least a "host" and a "user-agent" field and most often also an "accept-language" and "referer" field, which allows us to infer some information about the origin and destination of these requests. The most common "host" fields for prefix 1.1.1.0/24 and 1.2.3.0/24 are listed in Table IV.

The host field can contain either an explicit IP address or a fully qualified hostname, which should match the packet's destination address. At the time of our experiment no hostnames would legitimately resolve to an IP address in 1.1.1.0/24 or 1.2.3.0/24 space. Thus a hostname in the host field indicates either a misconfigured DNS or a proxy or gateway wrongly rewriting the destination address. We also found a few cases (0.04%) where the host field contained an address from all three prefixes of the IPv4 private address space, namely 10.0.0.0/8, 172.16.0.0/12 and 192.168.0.0/16. While the

"referer" field contents	#reqs	#Dest addr	#Src addr	#Dest hosts
http://1.1.1.1/info.html	2619350	1	262	1
http://1.1.1.1/info	771519	1	160	1
http://www.wittelsbuerger.de/	80319	5	543	5
http://kinoinfos.bplaced.net/	51777	4	2605	4
http://www.erodate.pl/pl/odebrane/	49573	5	836	5
http://www.facebook.com/?http://www .facebook.com/home.php?ref=home	32194	6	2228	121
http://www.facebook.com/?ref=home	27589	6	2005	79
http://www.wittelsbuerger.com/	22482	5	168	5
http://s1.4399.com:8080 /4399swf/upload_swf /ftp/20080421/6.swf	21397	1	348	1
http://1.1.1.40:8888/Openobject /down0.asp?id=1594	16445	1	241	2

TABLE V
THE 10 MOST COMMON "REFERER" FIELD VALUES FOUND IN HTTP REQUESTS TO PORTS 80, 81, 82, 8000, 8080 AND 8888 FOR PREFIX 1.1.1.0/24

1.1.1.0/24		1.2.3.0/24	
user-agent	count	user-agent	count
SyLink Profile Session	585577	x	466594
Mozilla/5.0	525040	SiteBacker	266945
MxPEG-ActiveX	464192	Mozilla/5.0 (compatible; Yahoo! Slurp/3.0)	216892
Microsoft URL Control - 6.00.8169	259060	Mozilla/4.0	136132
VCSoapClient	214131	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	132376
SeaPort/1.2	207069	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	101753
WinHttp-Autoproxy-Service/5.1	188000	Windows-Update-Agent	101346

TABLE VI
THE 7 MOST COMMON "USER-AGENT" FIELD VALUES FOUND IN HTTP REQUESTS TO PORTS 80, 81, 82, 8000, 8080 AND 8888

top 5 most common host field values in prefix 1.1.1.0/24 are explicit IP addresses matching the destination, there is a long tail to the distribution – 36.2% of all host fields contain a hostname, implying misconfiguration near the source (and in 1.2.3.0/24 this rises to 62.3% of all HTTP connections).

The host field may match the destination address for a number of reasons:

1) User explicitly enters an IP address into their browser
2) User unwittingly follows a link (embedded in a web page or provided by other software) which contains the IP address instead of a hostname
3) User gets redirected to a web page on a private 1.0.0.0/8 network with an IP address as URL (a common technique for Wi-Fi Hot-spots [13]), and the private network is leaking connections to the Internet.
4) An embedded HTTP client is establishing HTTP connections with or without the user's knowledge (such as software update clients, anti-virus and anti-spyware programs, download tools or malware)

Although possible, it seems unlikely that thousands of users are typing addresses like 1.1.1.1 and 1.2.3.4 in their browsers. Table V reveals additional information that may be obtained from the "referer" field of HTTP data heading to 1.1.1.0/24. The top two referer fields are a good example of Wi-Fi hot-spot login pages, such as provided by [13]. Together they are found in 23.8% of the HTTP connections.

Table VI shows the top 10 "user-agent" fields (where provided). The majority are not generated by well known browsers, but can be attributed to malware or software update services from various vendors. For example, user-agent "x" (1.2.3.0/24) is likely associated with malware, as it always occurs in combination with a host field of "chewinggym.com" (Table IV) which is used by the W32/Nugg worm [14]. Additionally "user-agent" values could be attributed to browsers, browser plug-ins, "browser toolbars", proxy clients, smart-phone browsers or smartphone applications and unknown applications or software.

We also analyzed the HTTP traffic for the values in the "proxy-connection" field. we found that it is set for 1.8% of the connections with a "referer" field for prefix 1.1.1.0/24 and for 0.44% of such connections for prefix 1.2.3.0/24, while it is set for approximately 3.8% for prefix 1.1.1.0/24 and 7.6% for prefix 1.2.3.0/24 if no "referer" field is present. If the "proxy-connection" field is set, and the "referer" field is missing, the "host" field value contains "microsoft.com" in 41.3% of the cases in 1.1.1.0/24, and "www.sinfors.com.cn" in 88.7% of the cases in 1.2.3.0/24. User agents for the cases in 1.1.1.0/24 are mostly "Microsoft URL Control - 6.00.8169" and "VCSoapClient". The user agents found in 1.1.1.0/24 belong to Microsoft's anti-phishing service for IE7 or IE8 [15]. It shows again hosts using misconfigured proxies, leaking private networks or misconfigured DNS resolvers. For connections to "www.sinfors.com.cn", we don't find any "user-agent" field, and the "proxy-connection" field is set to "close". The domain belongs to a company producing WAN optimization software and VPN software [16], thus we believe that we are dealing with a developer mistake in a client software of this manufacturer.

V. CONCLUSIONS

This paper extends previous studies of IPv4 traffic being sent to previously-unllocated 1.1.1.0/24 and 1.2.3.0/24 address space. We focus on TCP traffic during a brief period in mid-March 2010. By studying the reactions of sources when we respond to their TCP SYN packets with our own TCP SYN-ACK, we are able to assess the degree to which inbound traffic appears to be intentionally or unintentionally targeting destinations in 1.1.1.0/24 and 1.2.3.0/24.

Over a 10 hour period our simplified honeypot elicited 38.8M responses from 54.3M connection attempts to 1.1.1.0/24 and 10.5M responses from 13.5M connection attempts to 1.2.3.0/24 (representing 71% and 78% of all initial SYNs to 1.1.1.0/24 and 1.2.3.0/24 respectively). We believe many of the non-responsive sources come from asymmetric leakage of private networks using 1.0.0.0/8 space (internally destinated SYN packets instead leak outside through the NAT, but our returned SYN-ACKs are blocked).

By evaluating patterns of behavior by responsive sources we conclude that most sources attempting to establish a genuine TCP connection into 1.1.1.0/24 or 1.2.3.0/24 generally do so due to misconfiguration somewhere close to the source, rather than because the source application actively intends to probe destinations in 1.1.1.0/24 or 1.2.3.0/24 space. Close analysis of payloads from received HTTP-like connection attempts suggest that misconfiguration problems affect legitimate users, embedded clients (such as automatic software updaters), and malware alike. This suggests we might 'clean up' these prefixes through better education of the end-users and network equipment administrators.

VI. Acknowledgments

This work has been made possible in part by a grant from APNIC Pty Ltd.

References

[1] Internet Assigned Numbers Authority - IANA, "IANA IPv4 Address Space Registry." [Online]. Available: http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.txt

[2] RIPE NCC, "Pollution in 1/8," Feb. 2010. [Online]. Available: http://labs.ripe.net/content/pollution-18

[3] G. Huston and G. Michaelson, "Traffic in Network 1.0.0.0/8," Mar. 2010. [Online]. Available: http://www.potaroo.net/ispcol/2010-03/net1.html

[4] E. Wustrow, M. Karir, M. Bailey, F. Jahanian, and G. Huston, "Internet background radiation revisited," in *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 62–74.

[5] L. Spitzner, *Honeypots: Tracking Hackers*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[6] North American Network Operators Group – NANOG, "1/8 and 27/8 allocated to APNIC (Discussion)," Jan. 2010. [Online]. Available: http://mailman.nanog.org/pipermail/nanog/2010-January/017402.html

[7] Fonality, "trixbox - The Open Platform for Business Telephony." [Online]. Available: http://fonality.com/trixbox/

[8] A. Fei, G. Pei, R. Liu, and L. Zhang, "Measurements on delay and hop-count of the internet," jan 1998.

[9] G. Lyon, "NMAP Security Scanner." [Online]. Available: http://nmap.org

[10] Blizzard Entertainment, "Blizzard Support." [Online]. Available: http://us.blizzard.com/support/article.xml?locale=en_US&articleId=21015

[11] Micros Systems Inc., "Micros Kitchen Display Systems." [Online]. Available: http://www.micros.com/Products/RES/KitchenDisplaySystem/

[12] Vitalwerks Internet Solutions, "no-ip, The DNS service provider." [Online]. Available: http://www.no-ip.com/

[13] Cisco Systems Inc., *Cisco Wireless Control System Configuration Guide*. San Jose, CA, USA: Cisco Americas Headquarters, 2010.

[14] F-Secure, "Virus description: P2P-Worm:W32/Nugg." [Online]. Available: http://www.f-secure.com/v-descs/p2p-worm_w32_nugg.shtml

[15] Aqtronix, "User Agents Database." [Online]. Available: http://www.aqtronix.com/useragents/?Action=ShowAgentDetails&Name=VCSoapClient

[16] SANGFOR, "Sangfor Company Overview." [Online]. Available: http://www.sangfor.com/company/overview.html